



CleanStreets: Management of Urban Areas using a Mobile Application

Final Project Report

TU857

BSc (Hons.) in Computer Science (Infrastructure)

Ben Johnson

C20316733

Supervisor: Damian Gordon

School of Computer Science
Technological University, Dublin

12/04/2024

Abstract

This section will outline a concise summary of the overall project.

Dublin has become a very dirty city in recent years. A plague of incoherent road signage alongside cracked footpaths, lethargic traffic, overflowing bins and illegal dumping can be found on nearly every street across the city.

There are currently no ways for people who are in and around Dublin on a day-to-day basis to report these issues unless they want to raise a ticket online or make a call with the Dublin City Council. Between public transport stops, jobs, shops and key infrastructure for work, school, college, travel or visits, inhabitants and tourists alike are often forced to avoid rubbish on their journey. This project aims to create a solution to streamline the management and collection of this unnecessary scourge on the city streets.

The CleanStreets mobile application will provide a platform for the intelligent management of urban areas and collection of waste across Dublin. The project researches and develops a system that allows users to report issues from their locality, to help the relevant authorities (as well as community workers and volunteers) to address them. The project will focus on the use of usability and accessibility to improve the quality of the application development to increase the efficiency of clean-up operations and give everyday people the opportunity to address problems in their area.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

A handwritten signature in cursive script that reads "Ben Johnson". The signature is written in black ink and is positioned above a horizontal line.

Ben Johnson

12/04/2024

Acknowledgements

I would like to thank my project supervisor Damian Gordon, for his constant support, encouragement, & guidance in this project, as well as Paul Bourke, my second reader, for his feature suggestions throughout development.

Secondly, I would like to thank Patrick McNulty from the National Disability Authority & Richard Whelan, Administrative Officer at Waste Management Services Division for Dublin City Council for their help in the evaluation & input of the project.

I thank my family, friends and my TU857 classmates for their support, advice, and guidance with the research and development of my project.

Additionally, I thank Neuem Design for aiding in the icon creation, typography choices & logo design.

Table of Contents

1. Introduction.....	8
1.1. Project Background	8
1.2. Project Description	10
1.3. Project Aims and Objectives	12
1.4. Project Scope	12
1.5. Thesis Roadmap.....	13
2. Literature Review.....	14
2.1. Introduction.....	14
2.2. Alternative Existing Solutions to The Problem	14
2.3. Technologies researched	18
2.3.1. Software Tools	18
2.3.1.1. Programming & Scripting Languages.....	18
2.3.1.2. Operating Systems.....	21
2.3.1.3. Databases	23
2.3.2. Software Architectures	25
2.3.2.1. General Architecture Options.....	25
2.3.2.2. Specific Architecture Options	26
2.3.3. Algorithms	28
2.3.3.1. Routing.....	28
2.3.3.2. Mapping.....	29
2.3.3.3. Image Processing	29
2.4. Other Relevant Applications.....	30
2.4.1. PAC Waste Tracker	30
2.4.2. Holistic Trash Collection System.....	30
2.4.3. A Systematic Review of Mobile Apps for Waste Management.....	31
2.5. Existing Final Year Projects	32
2.6. Conclusions	33
3. CleanStreets System Design	34
3.1. Introduction.....	34
3.2. Software Methodology.....	34
3.3. System Architecture	36
3.3.1. Architecture Diagram	37
3.3.2. Prototype Design	37
3.3.3. Use Cases	42
3.4. Environment and User-Based Design Activities	44

3.4.1. Psychographic Exploration	44
3.4.2. Expert Consultation	46
3.4.3. Questionnaire	48
3.5. User Experience (UX) Considerations	50
3.5.1. Screen design.....	50
3.5.1.1. Nielsen’s F-shape.....	50
3.5.1.2. Principles of Universal Design	51
3.5.1.3. Theme	52
3.5.1.4. Graphic Design.....	53
3.6. Prototyping Software.....	55
3.6.1. Accessibility	55
3.6.1.1. Language Support.....	55
3.6.1.2. Font Size & Colours.....	55
3.6.1.3. Contrast	56
3.7. Conclusions	57
4. CleanStreets Development.....	58
4.1. Introduction.....	58
4.2. Developing the Prototype.....	58
4.2.1. Setting up the system	58
4.2.2. Report Creation	58
4.2.3. Account Management	59
4.2.4. Map Integration.....	61
4.2.4.1. Map Style	63
4.2.5. Routing.....	64
4.2.6. Noticeboard	66
4.2.7. Word filtering	67
4.2.8. Analytics.....	68
4.2.9. Administrative Role	69
4.2.10. Report upvotes	70
4.2.11. Post replies	70
4.3. System Running	71
4.4. Key Development Challenges	76
4.5. Conclusions	77
5. Testing and Evaluation	78
5.1. Introduction.....	78
5.2. Plan for Testing	78

5.2.1. Local Testing	78
5.3. Plan for Evaluation.....	80
5.3.1. User Evaluation.....	80
5.3.1.1. Students Learning with Communities TUD.....	80
5.3.1.2. Non-Expert Evaluation.....	80
5.3.2. Expert Evaluation.....	81
5.3.2.1. Dublin City Council.....	81
5.3.2.2. Accessibility Experts.....	81
5.3.3. Automated Evaluation	82
5.4. Key Evaluation Learnings	85
5.5. Conclusions	86
6. Conclusions and Future Work.....	87
6.1. Introduction.....	87
6.2. Issues and Risks	87
6.2.1. Social Good	87
6.2.2. Additional Challenges	88
6.2.3. Self-reflection & Criticism.....	89
6.3. Plans and Future Work	90
6.3.1. Gantt Chart	91
6.3.2. Further Development	92
6.4. Conclusions.....	93
Bibliography.....	95

1. Introduction

1.1. Project Background

“Dublin is a dirty, smelly, sticky old town once again... Nobody expects a city centre to be pristine; it is a populated and lived-in place after all. But I think it not being utterly gross to walk around should be at least an ambition.” (Mullally, 2023)



Figure 1: Illegal dumping in Dublin

The *CleanStreets* project aims to help Dublin become a litter-free, clean, sustainable city. There are exemplars of best practice around the world to model this aim upon, including Darmstadt, Germany (Knöll et al., 2019) where the streets are clean and there is an effective waste management system. By leveraging techniques like the travelling salesman algorithm, statistical analysis, and user feedback, this project aims to create a scalable solution that can be applied to larger cities in the future.

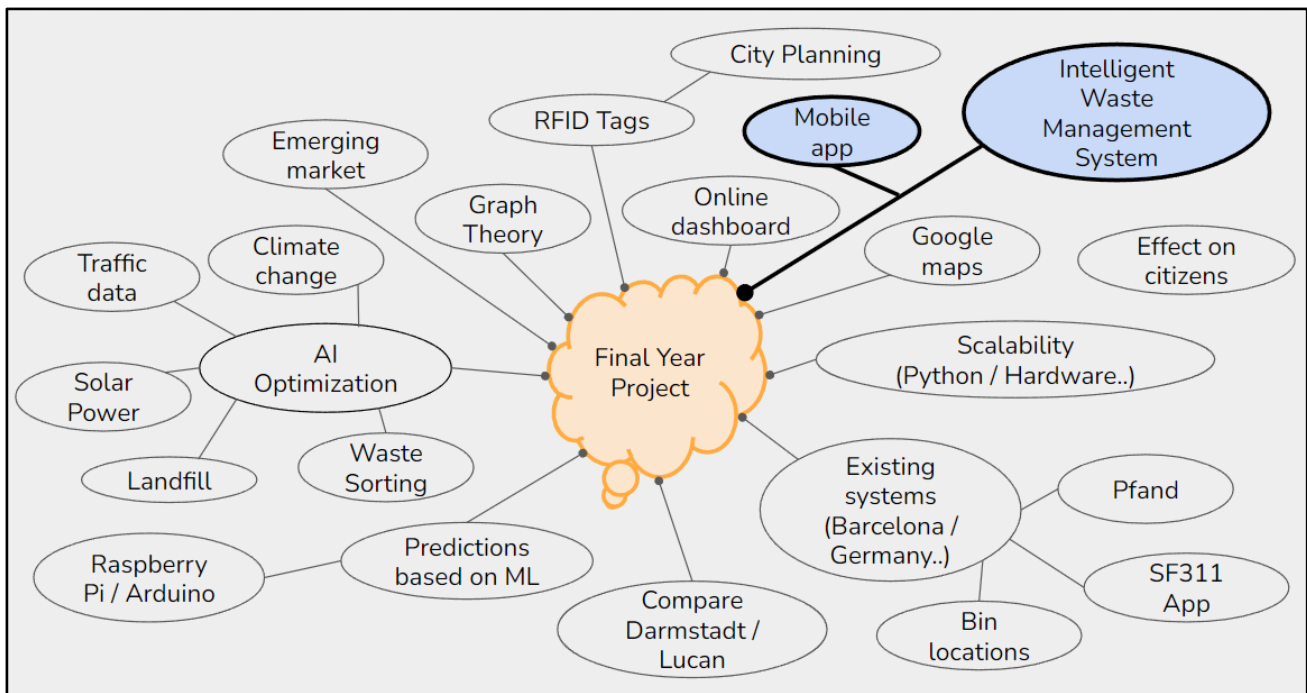


Figure 2: Mind map from the early stage of development

(Purcell & Magette, 2010) explored the attitudes and behaviour of people in the Dublin region towards waste management. They did this by developing a survey which they distributed to both commercial, non-commercial and residential sectors in three ways 100 businesses in different regions of Dublin were sent the survey in person, and 71 answers that were returned were deemed valid, 20 door-to-door interviews with residential address were undertaken, and a web-based survey resulted in 51 additional responses. The results from their survey indicated that the three different sectors (commercial, non-commercial and residential) responded in different ways, and therefore a series of targeted intervention strategies would lead to improved rates of waste management.

In 2021, (Hamilton & Mullarkey, 2021), alongside Smart City Programme Manager for Dublin City Council, Jamie Cudden, explored how technology can be utilised to address city challenges, to create a 'smarter' Dublin. They achieved this by reviewing recent city projects, such as the sensor-enabled smart gulley system, and talking to colleagues about various aspects of IoT systems. They discovered that a wide range of IoT projects have been tested across the city in recent years, yet there remained a low usage of such projects; caused by general lack of knowledge of IoT deployments & lack of awareness as to how they can be utilised effectively.

(Davies, 2007) highlights the gulf between semi-state waste management companies and the lack of societal waste policy improvement and development. The two main areas of waste-related civil society activities are examined, and the findings indicate that despite the differing strategies used by each group, some

conditions remain to congest the operation and development of future implementations.

1.2. Project Description

The *CleanStreets* waste management mobile application is focused on providing a user-friendly platform for reporting and addressing local environmental issues within Dublin. The application will primarily target Android devices, ensuring compatibility with the latest versions. The core functionalities of the app include enabling users to report various environmental problems they encounter, such as littering or illegal dumping, accessing real-time data related to these issues, and facilitating the process of addressing them efficiently.

The app will interact with a secure Firebase database, offering storage for user reports, allowing real-time updates, and ensuring the privacy and security of users' data. Key features within the application encompass a best route-finding algorithm that calculates optimal paths for issue resolution, a comprehensive statistics dashboard for users to gain insights into the local environmental landscape, and a notification system to provide timely updates on reported issues.

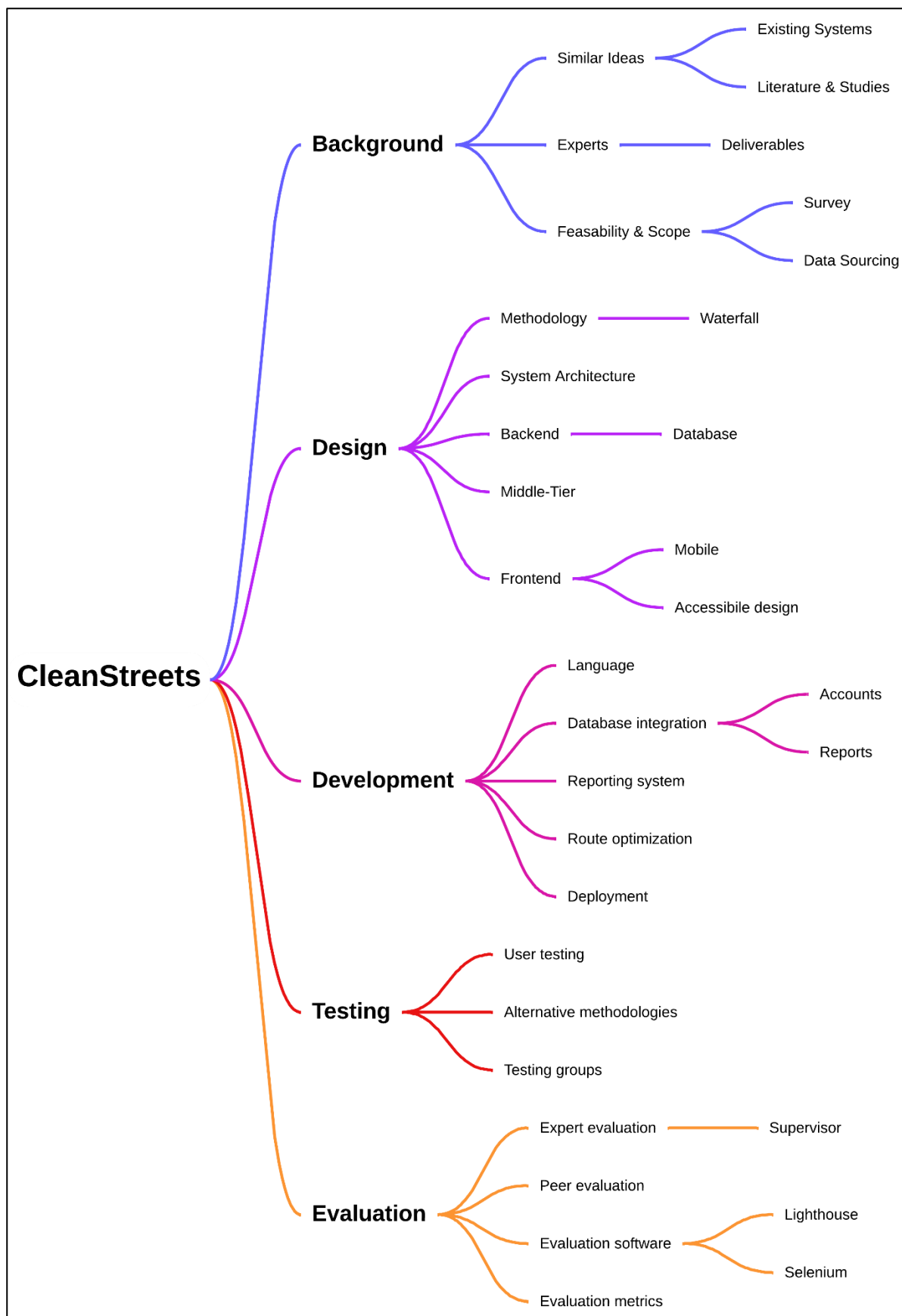


Figure 3: The overall project represented using branches

1.3. Project Aims and Objectives

The key aim of the *CleanStreets* project is to develop a mobile application that provides a user-friendly platform for reporting and addressing local environmental issues within Dublin. To achieve this aim, the following objectives need to be achieved:

- A user-friendly Android mobile application that allows users to report and access information about nearby issues in their locality.
- A secure and scalable Firebase database to store all the user reports and private data.
- A routing feature designed to output the best route between user-reported issues and plot the route on the map.
- Statistics dashboard for users, providing a deeper dive into the data of user reports in their area.
- A real time notification system that alerts users to updates / amendments to their own or their bookmarked reports.
- Comprehensive documentation including user guides, technical advice, installation instructions and project development.
- Accurate, responsive, and intuitive UI
- Thorough testing and quality assurance, including communication with councils and local groups.
- A universally accessible application, including extensive variable app settings, language support, font sizes, non-intrusive colours.
- Successful deployment

1.4. Project Scope

The *CleanStreets* project, is aimed at leveraging mobile technology to address environmental issues and promote cleaner, more sustainable urban living. This project revolves around the development of a user-friendly Android mobile application.

CleanStreets is designed for the community to report and tackle local environmental and quality of life concerns in urban areas across Dublin City. My primary focus is on creating an efficient and intuitive app for Android devices. While the concept of expanding to iOS is appealing, the scope remains constrained to Android due to my app development experience and to ensure the timely delivery of a robust application.

The core objectives of this project encompass user-generated issue reporting, an optimal route-finding algorithm, and efficient amendment of said issues.

CleanStreets does not extend to web-based or desktop applications. While machine learning technologies present exciting possibilities, they are beyond the current scope of development. It's designed to be a scalable and user-friendly solution, acknowledging the potential for future enhancements beyond this current phase.

1.5. Thesis Roadmap

Chapter 2 provides a literature review of some of the key research relevant to this topic, as well as a review of some existing projects and resources related to the development of this project. Additionally, a range of key technologies will be reviewed.

Chapter 3 is the Design chapter and outlines the project architecture, design methodology and systems involved. It includes Screen Prototypes, as well as Use Case Diagrams and Class Diagrams to discuss the design of the system.

Chapter 4 outlines the Prototype Development process, which outlines the development of the prototype of this system, including software architecture as well as programming languages and development environments.

Chapter 5 is the Testing and Evaluation chapter, which outlines the processes and methods used to test and evaluate my project. The testing process includes the development of test plans and unit testing, and the evaluation will look at UX and user-centred techniques.

Chapter 6 explains some of the Issues and Future Work part of the project, which will explore the risks and further development of the project should I build upon it in the future.

2. Literature Review

2.1. Introduction

In this chapter, the current landscape of ideas related to the *CleanStreets* project will be discussed, including an exploration of existing implementations of similar applications, as well as a review of some of the key technical, and domain-specific research that is relevant to this project.

2.2. Alternative Existing Solutions to The Problem

Several countries have been exploring approaches to ensuring efficient waste management processes, and well the development of specific technologies to co-ordinate and analyse waste trends. Below are outlines of some of the ones that will serve as inspiration for this project.

- **Prague, Czechia**

The smart bins in Prague feature sensors that will alert authorities if the bins are full or clogged, using a mobile application, “My Prague” (Smart Prague, n.d.). 464 sensors allow users to view detailed statistics from smart bins across the city. One report suggests that *“the annual savings for Prague 1 (a central district) alone will equal CZK 300,000”* (Hřib, 2019), or just over €12,000. *“Exactly these solutions fit my vision of a smart, data-driven city. It makes me happy. Thanks to the ICT Operator team and Smart Prague!”*, shared the mayor.

Benefits: Free and efficient application containing all information about everyday life in Prague, including waste management.

Relevant features: Accessible detailed statistics, lower investment cost, reduced carbon emissions, community rating system for existing amenities, faster than using a search engine.

Services: Simplified access to all Prague information from one location



Figure 4: Example of the MyPrague app (Hřib, 2019)

- **Singapore**

Singapore's Agency for Science, Technology, and Research (A*STAR) and a consortium of businesses have introduced BINgo, a smart waste bin that utilizes artificial intelligence, the Internet of Things, and smart sensors to improve recycling practices and increase recycling rates in Singapore. (Ocampo, 2022)

The bin interface features educational facts and tips about the importance of recycling. The data collected is used to identify the types of waste being thrown into the bins, as well as facilitate statistical analysis to optimize waste management.

Benefits: Increased recycling rates through education, AI-enabled technology streamlines the collection process

Relevant features: Data collection, user interface, AI integration

Services: User interface provides educational messages, recycling guidance



Figure 5: BINGo smart recycling bin in Singapore (Ocampo, 2022)

- **San Francisco, USA**

SF311 App for residents and visitors to San Francisco allows users to report quality of life issues quickly and easily by sending pictures, a brief description, and a map-based location. Requests are automatically routed to the relevant authority for amendment. (sf.gov, n.d.)

Benefits: Quicker issue resolution

Relevant features: Users can attach pictures, map-based location reporting

Services: Reports are automatically directed to the relevant authorities, Users can track the status of their submitted reports

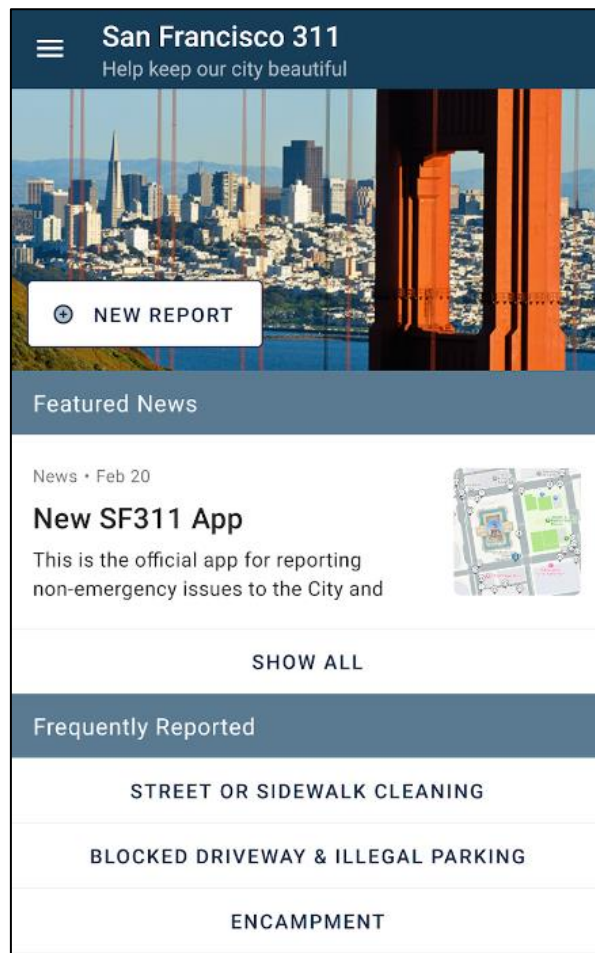


Figure 6: SF311 App for San Francisco

- **Latin America & The Caribbean**

GPS tools and remote surveillance cameras in waste collection trucks improved the efficiency of waste collection route control activities, strengthened operator transparency, and generated communication channels with end users.

RFDI (Radio Frequency Identification) chips offer access to real-time information on the volume and quantity of collected waste, contributing to the efficiency of waste collection services.

Geospatial data is gathered through AI-powered drones to monitor operations at final disposal sites (landfills and open dumps) and identify new sites for future facilities.

Implementing these systems increased operational efficiency by:

- Providing smart routing design.
- Improving operator productivity (by approximately 12%).

- Reducing costs through efficiency in fuel consumption and preventive maintenance.

(Guerra & Basani, 2023)

Benefits: GPS tools enables smart routing design, cheaper operational costs

Relevant features: Integration of GPS tools, AI-powered drones enable the collection of geospatial data

Services: Transparency with end users

2.3. Technologies researched

As part of the research for this project a few technologies were researched to explore what ones would be suitable for the development of this system, the three main areas of research were looking at Software Tools, System Architectures, and Algorithms that may be used.

2.3.1. Software Tools

This subsection outlines some of the key potential software tools that may be used as part of this project, looking at Programming and Scripting Languages, Operating Systems and Databases.

2.3.1.1. Programming & Scripting Languages

Some potential programming and scripting languages could be used in the implementation of this system, with each one offering its own advantages and disadvantages, some of which are outlined below.

- **JavaScript**



A fundamental technology in web-development. JavaScript is a high-level language that is often compiled '*just-in-time*'. It has dynamic typing, prototype-based object-orientation, and first-class functions (Flanagan, 2020). JavaScript is versatile, accommodating event-driven, functional, and imperative programming styles. It has APIs for handling text, dates, regular expressions, standard data structures, and the Document Object Model. JavaScript is supported by all major web browsers and is used extensively in client-side web development. A feature-rich set of libraries like React, Angular and Vue.js make development efficient and quick. Server-side deployment using Node.js is supported, as well as real-time app updating using the

built-in API. The JavaScript community is vast and active, so common problems can be dealt with quickly.

- **Python**

A high-level, yet relatively simple general purpose programming language (Lutz, 2013). The design of the language emphasizes code readability by use of indentations. Consistently ranked as one of the world's most popular languages, Python supports multiple standards, including structured, object-oriented, and functional programming. The term "*Swiss Army Knife*" comes to mind when considering Python as the language of choice, it has clean and simple syntax, it is supported cross-platform, contains extensive libraries, has web support through Django and Flask, and is the current go-to language for Machine Learning and AI applications, not to mention the incredible community behind it's open-source development.

- **HTML / CSS**

HyperText Markup Language and Cascading Style Sheets are used together to form the basic structure of documents displayed in a web browser (Grant, 2018). HTML is universally supported by web browsers, simple to read and debug, and seamlessly integrates with other web technologies, like MySQL through PHP. CSS provides a "*Separation of Concern*" between the content and presentation. CSS is also responsive to different screen sizes, devices, and software, as well as being cached by browsers to increase loading times on return visits. CSS also has an active community with many frameworks & libraries like Bootstrap and Foundation.

- **Java**

Java is a general-purpose programming language designed to let programmers write once, run anywhere (Gosling, 2005). Java is a high-level, class-based, object-oriented language that is designed to have as few implementation dependencies as possible. Once the Java Virtual Machine is installed, Java applications can be run on any machine regardless of the underlying architecture. It is versatile and suitable for web development or high-performance mobile application development, as modern versions of the JVM optimises the code execution. It contains built-in security features, a vast ecosystem of libraries, frameworks & tools, and is backed by a large, active group of developers and programmers.

- **Kotlin**

Kotlin is a versatile and powerful programming language known for its compatibility with Java and its concise syntax, thanks to type inference. It primarily targets the Java Virtual Machine, but it can also compile to JavaScript, making it suitable for frontend web applications like React, or native code via LLVM, which is helpful for developing iOS apps that share business logic with Android apps. In May 2019, Google announced Kotlin as its preferred language for Android app development, and since October 2017, Kotlin has been offered as an alternative to the standard Java compiler. (Lardinois, 2019)

- **C#**

C# is an adaptable and widely used high-level programming language that supports various programming paradigms, including static & strong typing, imperative, declarative, functional, generic, object-oriented, and component-oriented programming styles (Skeet, 2019). It is closely tied to the Microsoft ecosystem, making it suitable for Windows based applications and technologies like .NET, Azure, and Visual Studio. It allows for cross-platform development, has built-in security features, and is used extensively in the development of mobile apps and video games. It is enterprise-ready and scalable and is now a mature language; meaning it is more reliable and stable than others. Like many of the other programming languages that have been researched, C# has a rich library set and extensive documentation and guides through its active community of developers.

- **PHP**

PHP, or Hypertext Preprocessor, is a scripting language used to create web applications (Tuama, n.d.). PHP code is typically executed on a web server using a PHP interpreter, which can be implemented as a module, daemon, or Common Gateway Interface executable. PHP is not limited to web development and can be utilized for various programming tasks beyond the web, including standalone applications and even robotics. PHP has a relatively low learning curve and is similar in syntax to C & Java. There is an abundance of resources online, making it easy for developers like myself to work with. PHP enables rapid application development due to its simplicity and the availability of pre-built libraries and frameworks. It's well-suited for prototyping and getting web applications up and running quickly.

- **Swift** **Swift**

Swift is a powerful and intuitive programming language for all Apple platforms (Apple Inc., n.d.). It was created to support core concepts associated with Objective-C, such as dynamic dispatch, late binding, and extensible programming. However, it aims to do so in a safer manner, making it easier to catch software bugs and reducing common programming errors. Apple promoted "protocol-oriented programming" as a new paradigm of programming. It is currently the preferred choice for iOS and macOS development. It is integrated into XCode and is supported by open-source community contributions.

2.3.1.2. Operating Systems

Several potential operating systems can be used in the implementation of this system, depending on the deployment process. Some of the key and commonly used operating systems are outlined below.

- **Windows**

Windows is an adept and widely used operating system known for its user-friendly interface and widespread compatibility with various software applications (Adekotujo, 2020). It provides a stable and familiar environment for both desktop and server applications. Windows is closely integrated with the Microsoft ecosystem, making it a top choice for Windows-based applications and technologies like .NET, Azure, and Visual Studio. It supports a variety of programming languages and development tools, enabling developers to create a wide range of software solutions. Windows is recognized for its compatibility with a broad spectrum of hardware devices and peripherals, ensuring a seamless user experience. It is widely used in enterprise environments, thanks to its sound security features and support for business solutions. With a rich library of software and active developer community, Windows is a reliable and established platform for many diverse software development projects.

- **MacOS**

macOS is a flexible and user-friendly operating system developed by Apple (GCFGlobal, n.d.), known for its aesthetic design and seamless integration with Apple's ecosystem. It is the preferred platform for developing iOS and macOS applications through the XCode integrated development environment. macOS is also the primary environment for programming in Swift, Apple's modern programming language. It excels in areas like graphics, multimedia, and creative software

applications. macOS is recognized for its strong emphasis on user experience design, making it ideal for crafting visually appealing and intuitive software. The open-source community actively contributes to macOS development, and it is highly regarded for its security features and privacy controls. With a stable, Unix-like foundation, macOS is suitable for a variety of development projects, including web and mobile applications, among others.

- **Linux**

Linux is a resourceful and widely used OS which includes a large set of applications and other software components (Siever, 2009). It supports a range of programming paradigms, making it a preferred choice for both server and desktop applications. Linux is open-source and fosters a collaborative community of developers, allowing for extensive customization and integration with diverse technologies. It's closely associated with the open-source ecosystem and offers compatibility with numerous programming languages, development tools, and software frameworks. Linux is highly regarded for its stability, security, and reliability, making it a top choice for enterprise-grade solutions. Linux provides a solid foundation for a wide range of software projects. Its extensive documentation, active developer community, and plentiful library set make it a powerful choice for developers.

- **Android**

Android is a broadly accepted, Linux based (Meike, 2021) mobile operating system developed by Google. It powers a vast array of mobile devices, making it a dominant platform for mobile application development. Android's open-source nature and fruitful ecosystem of libraries and development tools make it a preferred choice for developers. It supports a wide variety of programming languages and development frameworks, enabling developers to create diverse and innovative mobile applications. Android is known for its flexibility, allowing developers to craft applications for smartphones, tablets, wearables, and even embedded systems. The Google Play Store provides a vast marketplace for distributing Android apps to a global audience. With a strong emphasis on security and privacy, Android ensures that user data remains protected. It is widely used for developing apps in Java, Kotlin, and other languages, and its extensive documentation and active developer community make it a reliable and accessible platform for mobile app development.

- **ios** **iOS**

iOS, developed by Apple, is a sophisticated and widely renowned mobile operating system that powers Apple's line-up of mobile devices, including iPhones, iPads, and iPods (Volle, 2023). It is celebrated for its elegant design, seamless integration with Apple's ecosystem, and focus on providing an exceptional user experience. iOS development is primarily programmed using XCode, Apple's integrated development environment, and Swift, Apple's modern and user-friendly programming language. iOS developers enjoy access to a vast range of development tools, libraries, and frameworks to create innovative and high-quality mobile applications. The Apple App Store serves as the primary distribution platform, allowing developers to reach a global audience. iOS is known for its security and privacy features, ensuring the protection of user data. The active iOS developer community and extensive documentation make it a reliable platform for mobile app development.

2.3.1.3. Databases

A few potential databases could be used in the implementation of this system, with each one offering its own advantages and disadvantages, some of which are outlined below.

- **PostgreSQL**

PostgreSQL, the open-source relational database management system (RDBMS), is noted for its versatility and concentrated design. It is a highly stable database management system, backed by more than 20 years of community development which has contributed to its high levels of resilience, integrity, and correctness (Amazon AWS, n.d.). PostgreSQL's support for multiple data types, indexing, and powerful queries makes it a prime choice for projects necessitating structured, high-performance data storage. With ACID compliance, PostgreSQL guarantees data integrity, while its extensibility empowers developers to craft custom functions and data types. An active developer community continuously enhances the system's features and reliability. PostgreSQL's versatility extends to its compatibility with various programming languages and support for geospatial data and JSON, rendering it a valuable option for diverse projects, from web applications to enterprise solutions and geospatial and data analytics applications.

- **MongoDB** **mongoDB**

MongoDB, a widely embraced NoSQL database, offers unparalleled flexibility and scalability for contemporary project development. As a document-oriented

database, MongoDB is a reliable, high-performance system which allows for almost infinite horizontal scalability (Chodorow, 2013). Its dynamic schema simplifies adapting data models, reducing development complexity. Using BSON (Binary JSON), MongoDB's document-based data model suits real-time, content-rich, and big data applications. Querying data with JSON queries is made easy by using the MongoDB Query Language. MongoDB's active community and comprehensive documentation provide abundant resources for developers. As an open-source solution compatible with various programming languages and frameworks, MongoDB is an appealing choice for modern, dynamic software projects.

- **Oracle Database**

Oracle Database is a powerful relational database management system, renowned for its enterprise-level capabilities. It excels in database programming, advanced SQL operations, OOP, and query optimization (Greenwald, 2013). With ACID compliance, it ensures data integrity. Its features include partitioning, clustering, and parallel processing, making it ideal for large-scale and high-transaction projects. Oracle Database is favoured for its reliability and security. It enjoys a mature ecosystem with strong developer support and comprehensive documentation. Compatibility with various languages and frameworks adds to its appeal, and integration with Oracle Cloud suits cloud-based projects. Oracle Database is particularly suited for data-intensive projects, where security, scalability, and reliability are paramount. Its extensive toolset for backup, recovery, and data analytics enhances its reputation as a top-tier database solution for demanding projects.

- **Firestore**

Firebase is a widely adopted mobile and web application development platform. It offers a broad set of services that streamline the development process, including things like analytics, authentication, databases, configuration & file storage (Stevenson, 2018). Firebase's real-time database is a standout feature, allowing developers to create responsive, data-driven applications. It simplifies user authentication and security, enabling secure access control. Firebase Cloud Functions enable serverless computing, while Firebase Hosting offers scalable web hosting. One of Firebase's strengths is its seamless integration with client-side libraries and frameworks, making it a premium choice for front-end development. Firebase's analytics and crash reporting provide valuable insights into user behaviour and application performance. Its scalability and real-time capabilities are

well-suited for mobile and web applications, especially for start-ups and projects with rapid development timelines. The platform's extensive documentation and active community further enhance its appeal for application development.

2.3.2. Software Architectures

This subsection outlines some of the key potential software architectures that may be used as part of the *CleanStreets* project, looking at first the general architecture options, and then more specific options.

2.3.2.1. General Architecture Options

In terms of a general software architecture design, the options for development include the following: 1-Tier, 2-Tier, 3-Tier, and N-Tier (Fowler, 2012). Each of these general potential architecture options are outlined below.

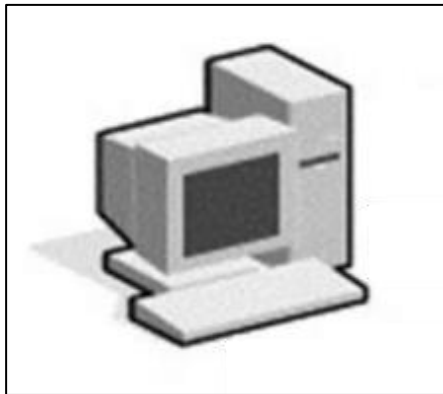


Figure 7: 1-Tier Architecture

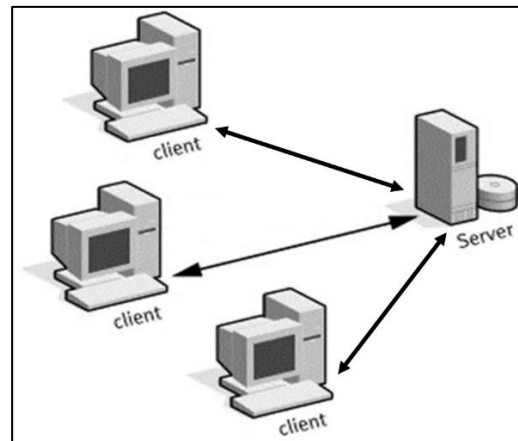


Figure 8: 2-Tier Architecture

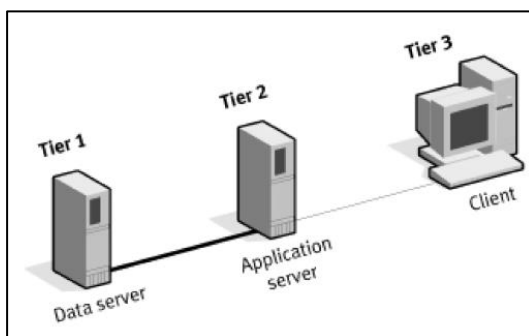


Figure 9: 3-Tier Architecture

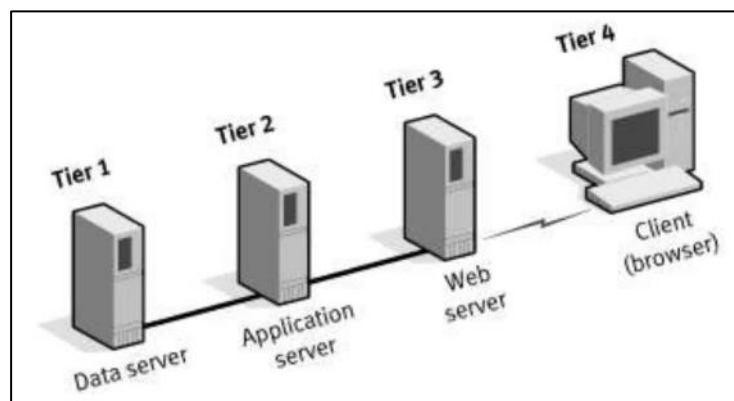


Figure 10: N-Tier Architecture

- **1-Tier**

In 1-Tier architecture, the entire application, including the user interface, business logic, and data storage, resides on a single machine. This approach is

simple but lacks scalability and flexibility. It's suitable for standalone desktop applications but is less ideal for complex, distributed systems.

- **2-Tier**

2-Tier architecture, also known as client-server architecture, separates the application into two layers: the client, which handles the user interface, and the server, which manages the data and business logic. This model is common in database-driven applications, where the client communicates with the server for data retrieval and manipulation.

- **3-Tier**

3-Tier architecture further divides the application into three layers: presentation (user interface), application logic, and data storage. This separation enhances scalability, maintainability, and security. The client interacts with the presentation layer, which communicates with the application layer, which in turn interacts with the data layer. This architecture is prevalent in web applications.

- **N-Tier**

N-Tier architecture is a modular and scalable approach that divides an application into multiple layers or tiers. It allows for greater flexibility and scalability by distributing specific functionalities across different tiers. Each tier serves a specific purpose, such as presentation, business logic, data access, or services. This architecture is suitable for large and complex systems, providing a high degree of organization, maintainability, and resource scalability.

2.3.2.2. Specific Architecture Options

In terms of specific software architecture design, the options depend on the operating system, programming languages and databases used. Some potential architecture options are outlined below.

- **LAMP**

A LAMP stack is a bundle of four different software technologies that developers use to build websites and web applications. LAMP is an acronym for the OS, Linux; the server, Apache; the database, MySQL; and the programming language, PHP (Amazon, n.d.). This stack is widely used for developing dynamic web applications and websites. Linux provides a stable operating system foundation, Apache serves as the web server, MySQL manages the database, and PHP handles server-side scripting. LAMP is renowned for its open-source nature,

affordability, and overall completeness, making it a preferred choice for web developers, especially those working on Linux-based servers. It offers a solid platform for creating web applications, from small websites to large-scale web services.

- **WAMP**

The WAMP stack is a Windows-based web development environment that is an alternative to the LAMP stack. WAMP stands for Windows, Apache, MySQL, and PHP. This stack is designed for Windows users, providing them with a comprehensive platform for web development. Windows serves as the operating system, Apache handles web server functions, MySQL manages databases, and PHP takes care of server-side scripting. WAMP is popular for its ease of setup and configuration on Windows machines, making it accessible to developers working in Windows environments. It offers the tools needed for developing web applications, from simple websites to complex web solutions.

- **XAMPP**

XAMPP is a cross-platform web development stack that simplifies the setup of a local development environment. XAMPP stands for cross-platform (X), Apache, MariaDB, PHP, and Perl. XAMPP is known for its ease of use and portability, making it a valuable tool for developers to set up a local web server environment on various operating systems, including Windows, macOS, and Linux. It provides the essential components for web development, allowing developers to work on their projects locally before deploying them to production servers. XAMPP is favoured for its simplicity and is an excellent choice for individuals and small teams looking for a hassle-free web development environment.

- **MAMP**

The MAMP stack is tailored for macOS users, offering a local web development environment on Apple's operating system. MAMP stands for macOS, Apache, MySQL, and PHP. It provides macOS users with a user-friendly solution for setting up a local web server environment, which is essential for web development and testing. macOS serves as the operating system, Apache handles web server functions, MySQL manages databases, and PHP takes care of server-side scripting. MAMP simplifies the process of creating, testing, and deploying web

applications for macOS users, making it a popular choice among developers and designers who work on Apple machines. It offers the necessary tools for local web development and project testing before going live.

2.3.3. Algorithms

This subsection outlines some of the key potential algorithms that may be used as part of the *CleanStreets* project, looking at Routing Algorithms, Mapping Algorithms, and Image Processing Libraries.

2.3.3.1. Routing

One aspect of the project is to identify sites where waste is reported, and if there are numerous reports, a routing algorithm may be needed to suggest the shortest route to visit all the reported sites.

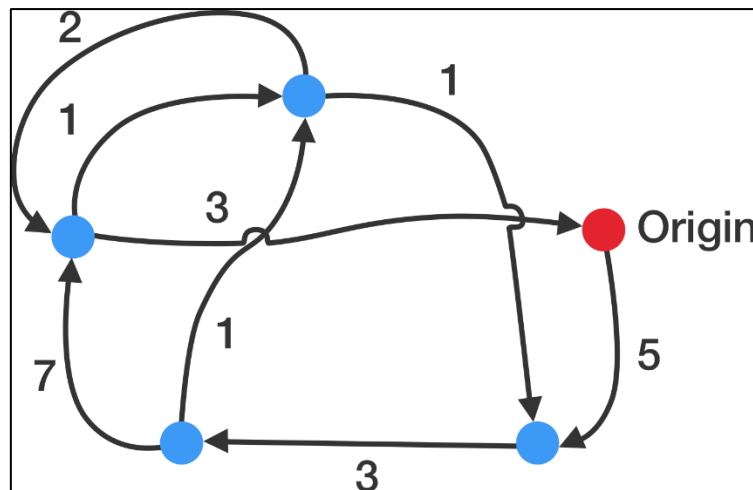


Figure 11: Routing example (Brilliant, n.d.)

- **Dijkstra's algorithm**

Dijkstra's algorithm is used to find the shortest path between two given nodes on a graph (Navone, 2020). The principle of the algorithm is to apply 'weight' to each edge, and process the path of least weight (i.e., shortest path). It has the potential to be used in map routing as the X, Y coordinates can be used as the nodes, and the routes, traffic, blockages can apply weight to the different edges.

- **A* search algorithm**

The A* Search algorithm searches for the shortest path between the initial and final state of an implementation, like graphs and maps (Chatterjee, 2023). It will calculate the cost of moving between areas (cells) until the shortest path to the destination is found. It is used in various applications, such as web mapping.

- **Google Directions API**



Google Directions API is a service that processes HTTP requests to return JSON or XML-formatted directions between two locations on a map (Google, n.d.). It allows for directions services for several modes of transportation, including transit, driving, walking, or bicycling, and is easily integrated into web and mobile applications.

2.3.3.2. Mapping

Another important aspect of the *CleanStreets* system will be to show the sites where waste or dangerous materials are reported, therefore two potential mapping tools are discussed below.

- **Google Maps API**



Offers developers the opportunity to display interactive maps and customize them however is needed for any project. Developers can leverage maps to help users create itineraries, for example (Juviler, 2022). Google Maps data refreshes in real-time, meaning that maps created with the Google Maps API will always be up to date for users. Like the Directions API, it is also easily integrated into web and mobile applications.

- **OpenStreetMap**



OpenStreetMap is powered by open-source software from its “slippy-map” interface to the underlying data access API. It is a free, editable map of the whole world that is being built by volunteers largely from scratch and released with an open-content license. The API allows for simple integration with web and mobile applications and projects. (OpenStreetMap, n.d.)

2.3.3.3. Image Processing

Another potential feature of the *CleanStreets* system is to allow the users to take pictures of the waste, and to do some image processing to ensure that the images do not violate privacy laws or GDPR, by blurring faces and house numbers.

- **Google ML Kit**



The Google ML Kit is a mobile SDK that provides a platform for on-device machine learning functionality to Android and iOS devices. It is a simple yet powerful API for many integrated features, such as Face Detection, Selfie Segmentation, Image Labelling, and more. (Google ML Kit, n.d.)

- **OpenCV**

OpenCV is an open-source computer vision and machine learning software library. It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. (OpenCV, n.d.)

- **TensorFlow Lite**
TensorFlow Lite

TensorFlow Lite is an open-source framework to run machine learning models on mobile and edge devices. It is suitable for image classification, object detection, speech recognition, natural language tasks, and more. It has support for Android, iOS, Linux, and embedded microcontrollers. (TensorFlow, n.d.)

2.4. Other Relevant Applications

2.4.1. PAC Waste Tracker

(Nagendra & Agarwal, 2019) discuss the development of a Waste Management App for Bangalore in India, a country that has witnessed the largest increase in urban population in the past four decades. This growth poses a number of challenges in terms of meeting the demand for key services, especially waste management. Before the development of this app only 20% of waste was picked regularly. To address this the Public Affairs Centre launched the PAC Waste Tracker app to allow citizen to report waste problems to allow the Council to identify the various issues plaguing the waste collection mechanism in the city with inclusivity. The app showed the many challenges associated with waste, including irregular waste collection, non-segregation of waste, inconvenience caused during collection, and non-compliance with transportation of waste. The findings of the study were presented to local councillors, service providers, and municipal bodies.

2.4.2. Holistic Trash Collection System

(Saher, Saleh, & Anjum, 2023) discuss the development of the HTC system (Holistic Trash Collection System) for Saudi Arabia. The system uses an internet of things (IoT) waste management system that aids remote waste level monitoring by using citizens who can report problems with waste collection by scanning QR codes affixed to waste bins. They can also report waste levels in the bins and transmit this data to the waste collection teams' database, allowing the teams to plan for optimized waste collection procedures, considering parameters such as waste volume and the most efficient collection routes aimed at minimizing both time and fuel consumption. This approach emphasizes the need for global commitment by

policymakers, stakeholders, and civil society, working together to achieve a common goal.

2.4.3. A Systematic Review of Mobile Apps for Waste Management

(Suruliraj, Nkwo, & Orji, 2020) undertook a systematic literature review of mobile apps for waste management with the main aim of focusing on the persuasive strategies employed in these apps to achieve specific behaviour. Specifically, they investigated 125 mobile apps for waste management and identified distinct persuasive strategies, including the following techniques (Oinas-Kukkonen & Harjumaa, 2008):

- The reduction strategy encourages users to perform target behaviours by breaking down complex activities into simple tasks.
- The tunnelling strategy encourages users to carry out target behaviours by guiding them through a process or experience during system use, hence reducing deviations.
- The tailoring strategy encourages users to carry out target behaviours by producing information displays relevant to the potential interests, context and other factors.
- The personalization strategy encourages users to carry out target behaviours by producing information displays relevant to the individual citizen.
- The self-monitoring strategy encourages users to achieve target behaviours by helping them to keep track of their individual performances or status along the way.
- The simulation strategy encourages users by helping them to see the relationship between cause-and-effect and their behaviours.
- The rehearsal strategy encourages users to rehearse a target behaviour before performing it in the real world.

Their results show that the apps cumulatively employed 251 persuasive strategies (with many incorporating more than one strategy) spread across the seven distinct primary task support persuasive strategies as follows: reduction (n=76), tunnelling (n=9), tailoring (n=37), personalization (n=75), self-monitoring (n=31), simulation (n=7) and rehearsal (n=16).

2.5. Existing Final Year Projects

Patrick Cummins: Ireland Neighbourhood Watch Application for Mobile Devices

This project is a mobile application designed to keep the public aware of any suspicious or dangerous activities within their area, using a reporting system. The application features an account-based system, where user information such as their location are stored in a database. The users have several choices after logging in; a button to display the map of reports in their home area, extrapolated by their location provided earlier, a button to add a report to the database, a secondary map, of Ireland, where statistics for that area including crimes and car crashes are shown. Reports are displayed on the map as markers and additional information is provided if the markers are selected.

This system had a large influence on the initial design and concept of CleanStreets. The key features are implemented in such a way that makes the app effortless to use. It is a great example of well implemented features, a great user experience and effective design. Since accessibility is one of the aims of CleanStreets, the user experience will be something to be mindful of throughout the frontend implementation.

Paul McKenna: Transport Dublin Android Mobile Application

This final year project is also a mobile application, which allows a user to find out about the different modes of public transport available in Dublin that can take them to the destination of their choice. The project is from 2014, but the core functionality and design are good inspiration for the CleanStreets application. The app utilises user GPS location to plot the nearest transport stops, which can help the user get to their destination. It uses real-time and timetable data from various sources to calculate and display to the user the expected time of next arrival and any additional information needed to identify the transport.

The project uses a client-server model, Android frontend and a geospatial MySQL database for storing and running the SQL commands. The available data is taken from Dublinked, a collection of open data for Dublin. The technologies used mirror CleanStreets, and the research and design are well documented and may prove very useful in future development.

2.6. Conclusions

This chapter provides an overview of some of the key research undertaken to help inform the design of the CleanStreets project. It begins with a list of some key international projects that serve as an inspiration for this project. Following that, some key potential technologies are presented, including some programming languages, some operating systems, some databases, etc. Following this, some existing final year projects that are relevant to this research are outlined.

3. CleanStreets System Design

3.1. Introduction

In this chapter a brief outline of some of the key design details will be outlined. First, a description of the Software Methodology used in this project will be described, followed by an outline of the System Architecture of this work. Next, a detailed description (with diagrams) of the front-end, middle-tier and back-end of the system will be presented. This is followed by a description of an initial set of experiments undertaken, including a walkabout, an interview and a questionnaire deployment.

3.2. Software Methodology

There were several software methodologies that suited the *CleanStreets* application, they are outlined below:

The Waterfall Methodology is a linear sequential design process, originating in software development processes. The waterfall development method originates in the manufacturing and construction industries. This Waterfall approach worked well for many IT projects because they tended to be tightly scoped in both time and cost, with relatively fixed requirements that did not change much during the project. Projects were small enough that changes could easily be managed by simply adding an extra week or two at either end of the project timeline without causing any real problems. For my application, the “Waterfall” methodology is used, as it follows the ideology behind the method. (Institute Project Management, 2022) (Royce, 1970)

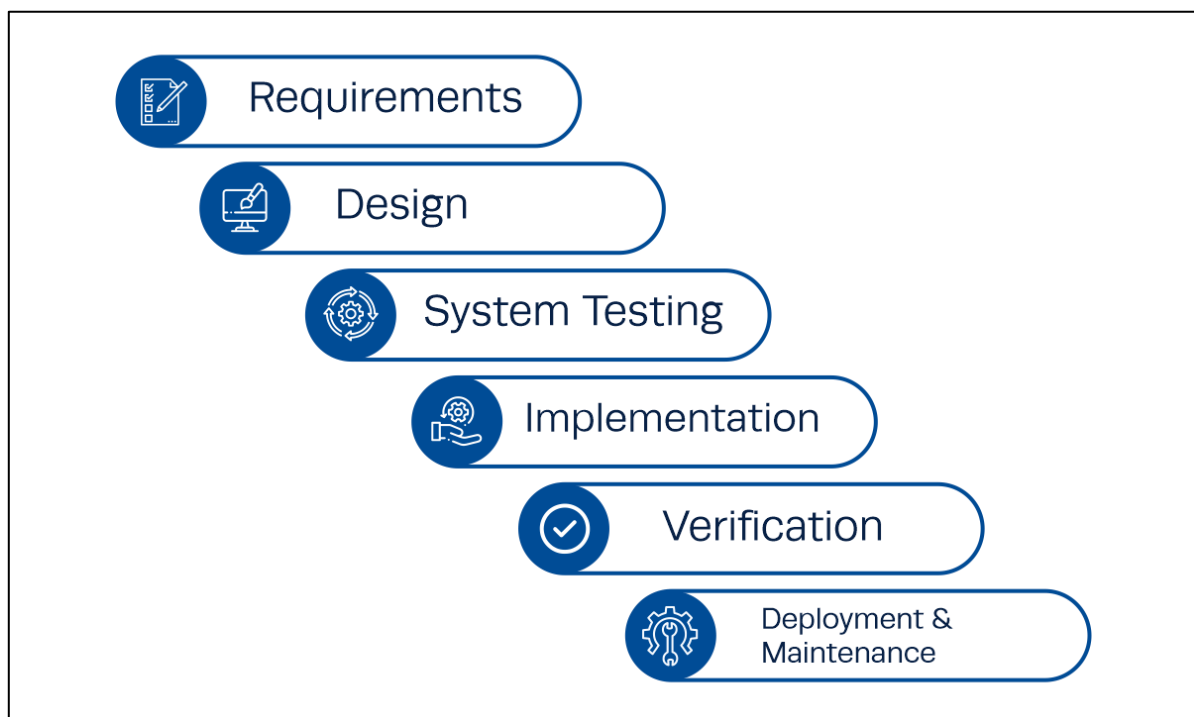


Figure 12: The Waterfall Methodology diagram.

The Spiral Model is a development lifecycle model that combines aspects of the Waterfall model with the iterative development process. Each loop around the figurative “spiral” represents a phase in the entire development of the project. The model allows for incremental release and refinement of the product, including prototype creation. The process is as follows: Determine objectives of the project, identify risks involved in the development, begin creating the product and perform testing, and finally plan the next phase of the project. (TechTarget, n.d.) (Boehm, 1988)

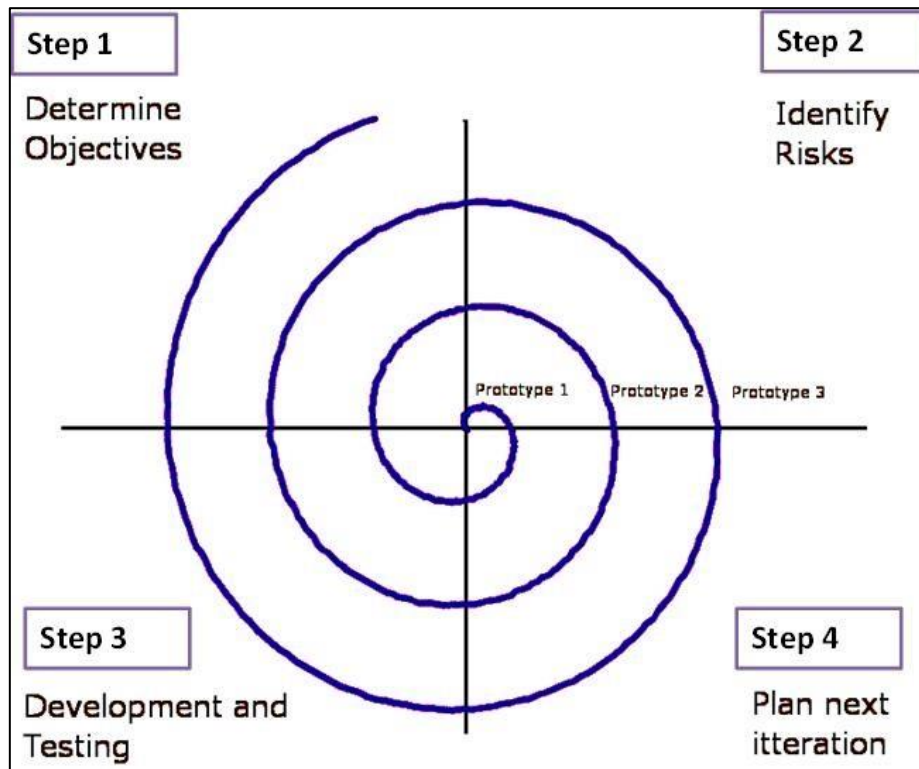


Figure 13: The Spiral Model (vtestcorp.com, n.d.)

Feature-driven Development, a model that is part of the Agile Approach, and it focuses on the small, client-valued requirements of a development. It involves initial project-wide activities (Developing overall model, build a list of features, plan by feature), then the iteration cycle of designing and subsequently building by feature. One of the final steps is to review the design. This approach decreases the chances of bugs and shortens the overall development cycle, as well as offering a well-

defined progress tracking and reporting capabilities. (Rychlý & Tichá, 2008)



Figure 14: Feature-driven Development lifecycle (ProductPlan.com, n.d.)

Everything considered, the **Feature-driven Development** model is the software methodology that best suits the development of the CleanStreets application, because it makes the developer divide the system into a few small tasks that can be implemented independently and when time permits.

3.3. System Architecture

This section discusses the overview of the system architecture. The functionality and intended implementation will be explained using Three-Tier Architecture, shown in figure 15.

3.3.1. Architecture Diagram

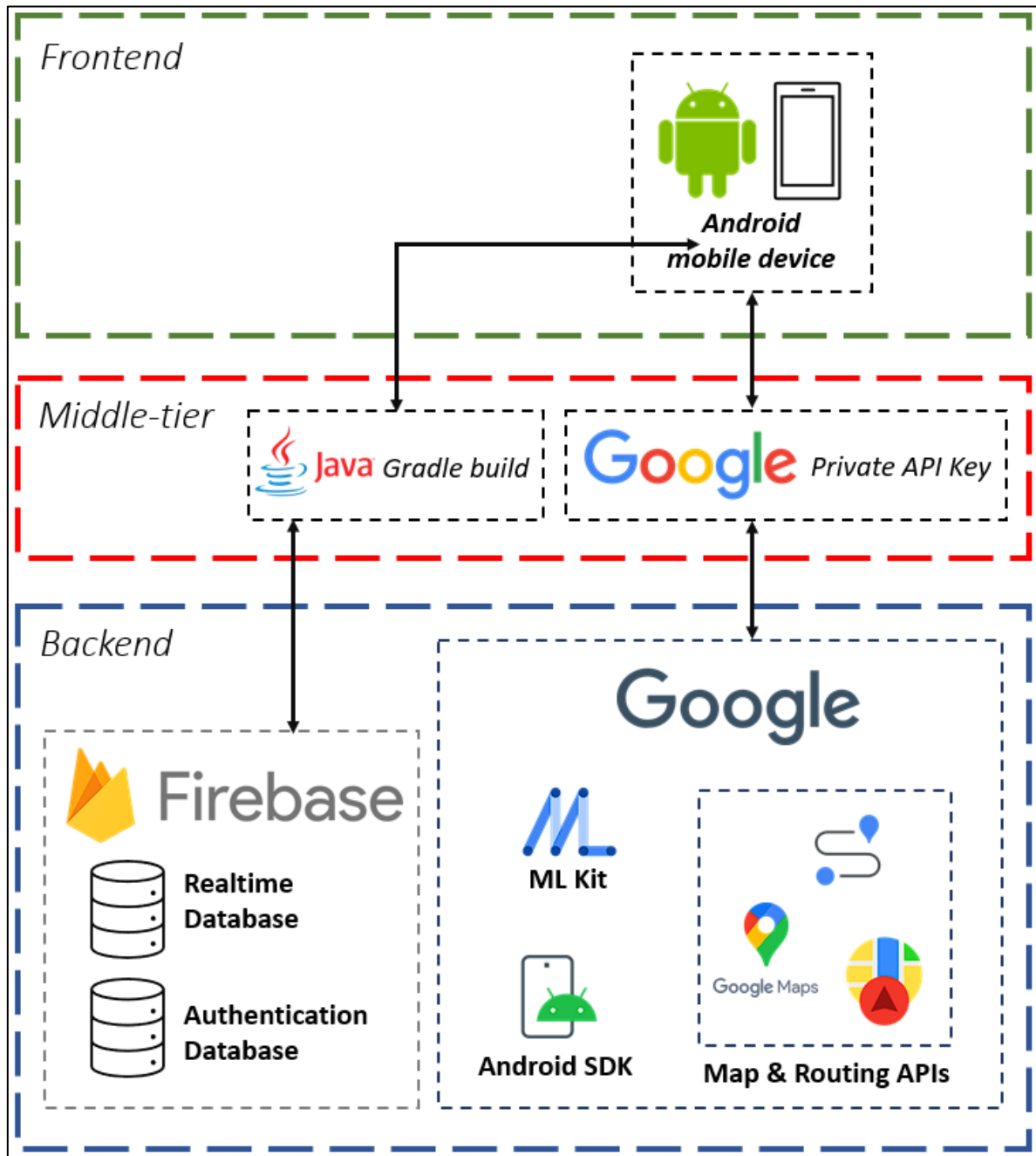


Figure 15: Software Architecture of the system.

3.3.2. Prototype Design

This section will present the paper mock-ups created to demonstrate the early design iterations of the *CleanStreets* application (Snyder, 2003). Each mock-up was created using a modern phone screen print-out, a ruler and a pencil.

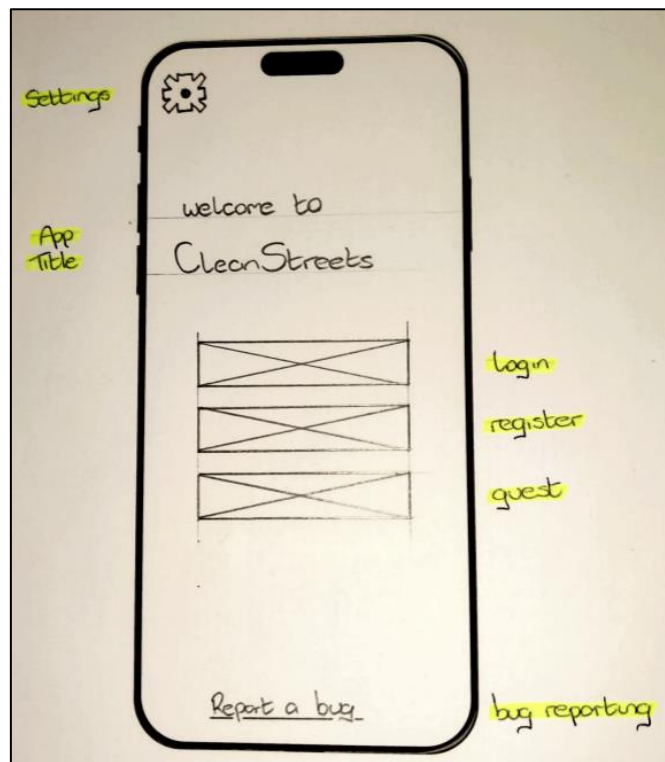


Figure 16: Paper prototype of the Main Screen

Figure 16 represents the screen that will be shown to users upon first opening the application. It includes a title, 4 buttons that will navigate the user around the app, and a link for reporting bugs.

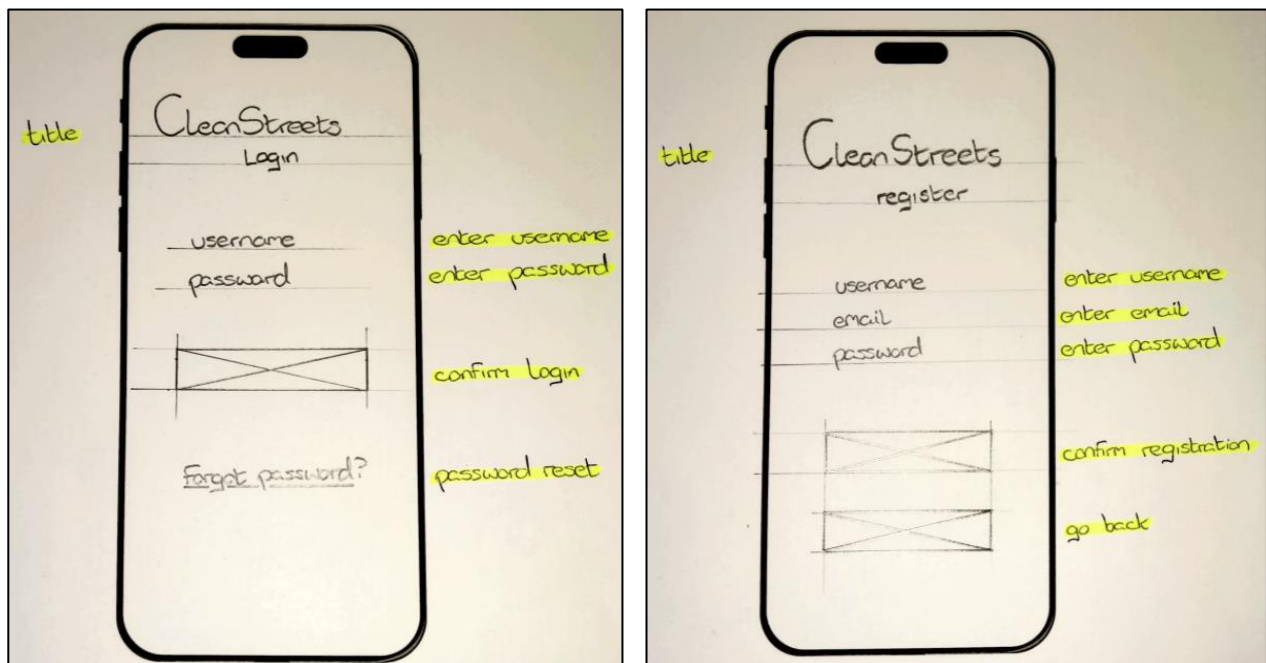


Figure 17: Paper prototype of the Login & Registration Screens

Figure 17 shows the Login & Registration screens as they appear in the app. The device back button can be used to return to the main screen. The login screen

features a “forgot password” link which will allow the user to reset their password. All the new registered users’ details are saved in the Firebase Authentication Database.

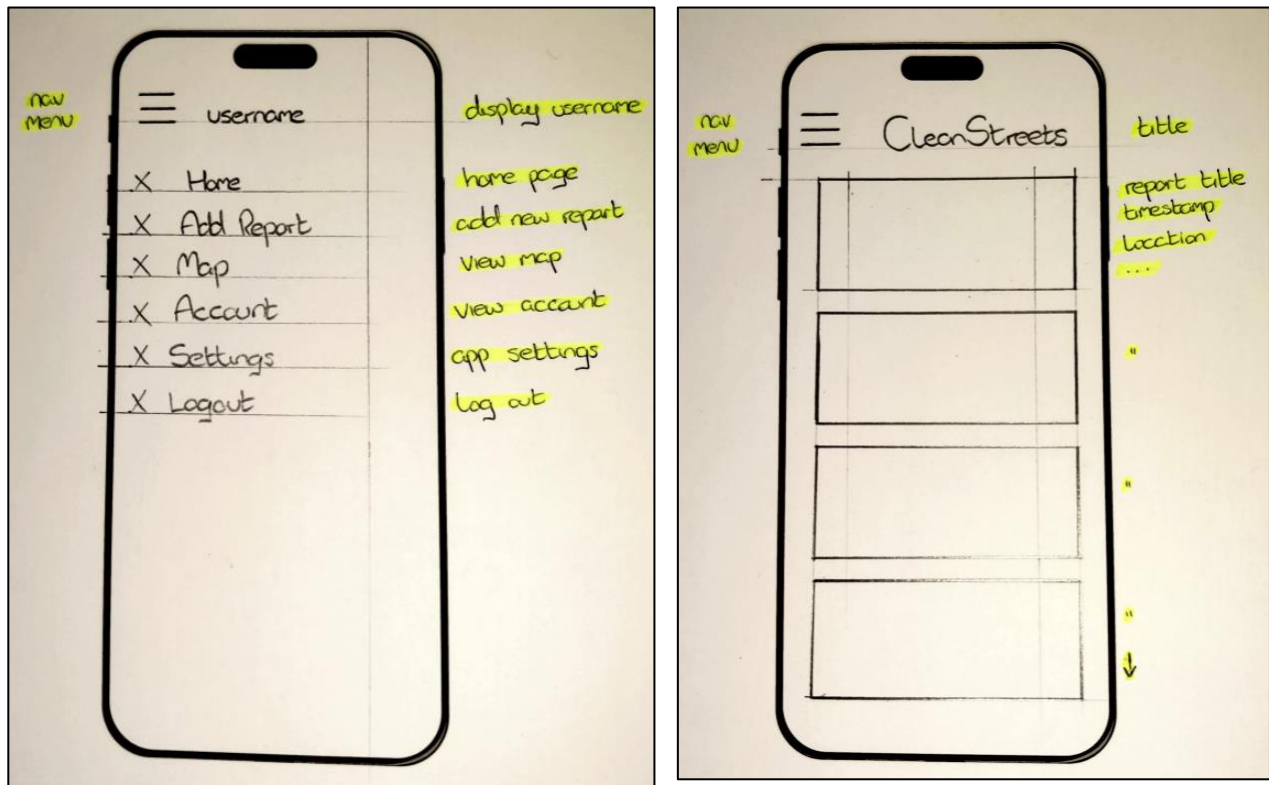


Figure 18: Paper prototype of the Navigation Bar & Report List screen.

Figure 18 shows a simple navigation bar & the list of reports screen, with links to all the other relevant areas of the application. Some buttons may be hidden depending on the user login status. Users may also click each report to get a more detailed view of the report.

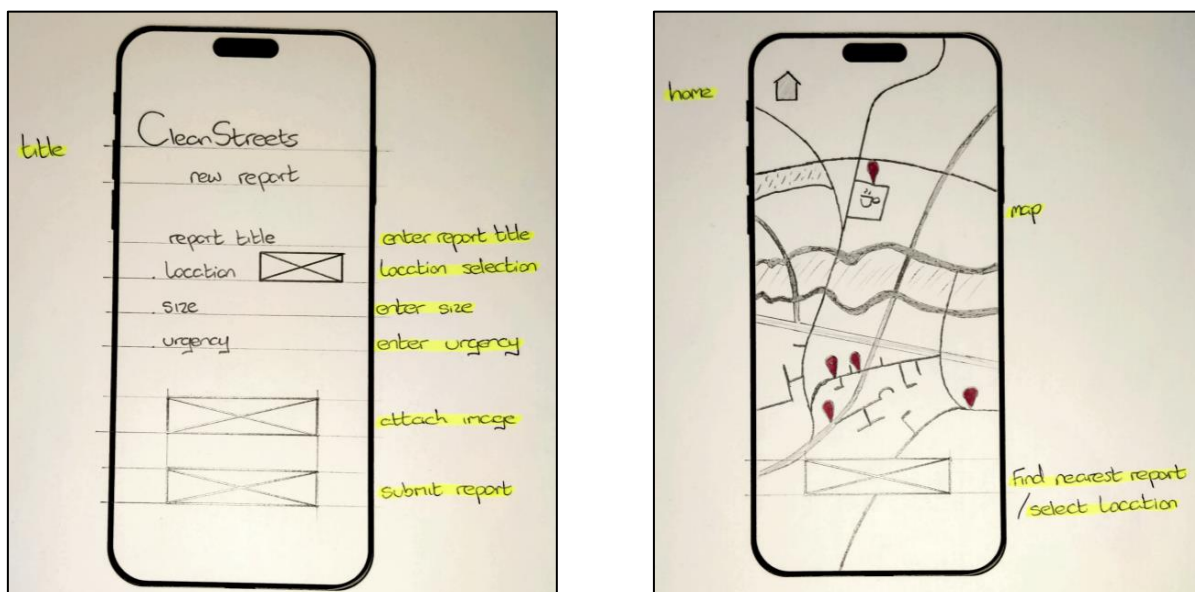


Figure 19: Paper prototype of the New Report Creation & Map Screens

Figure 19 represents the new report creation screen, as well as the map screen. Users supply details of their issue, including selecting the location from a map and attaching an image. Every report is saved in the Firebase Realtime Database. The project utilises maps in multiple ways, like finding the best route to the nearest report, selecting the location of a new report and displaying the reports as pins on the map.

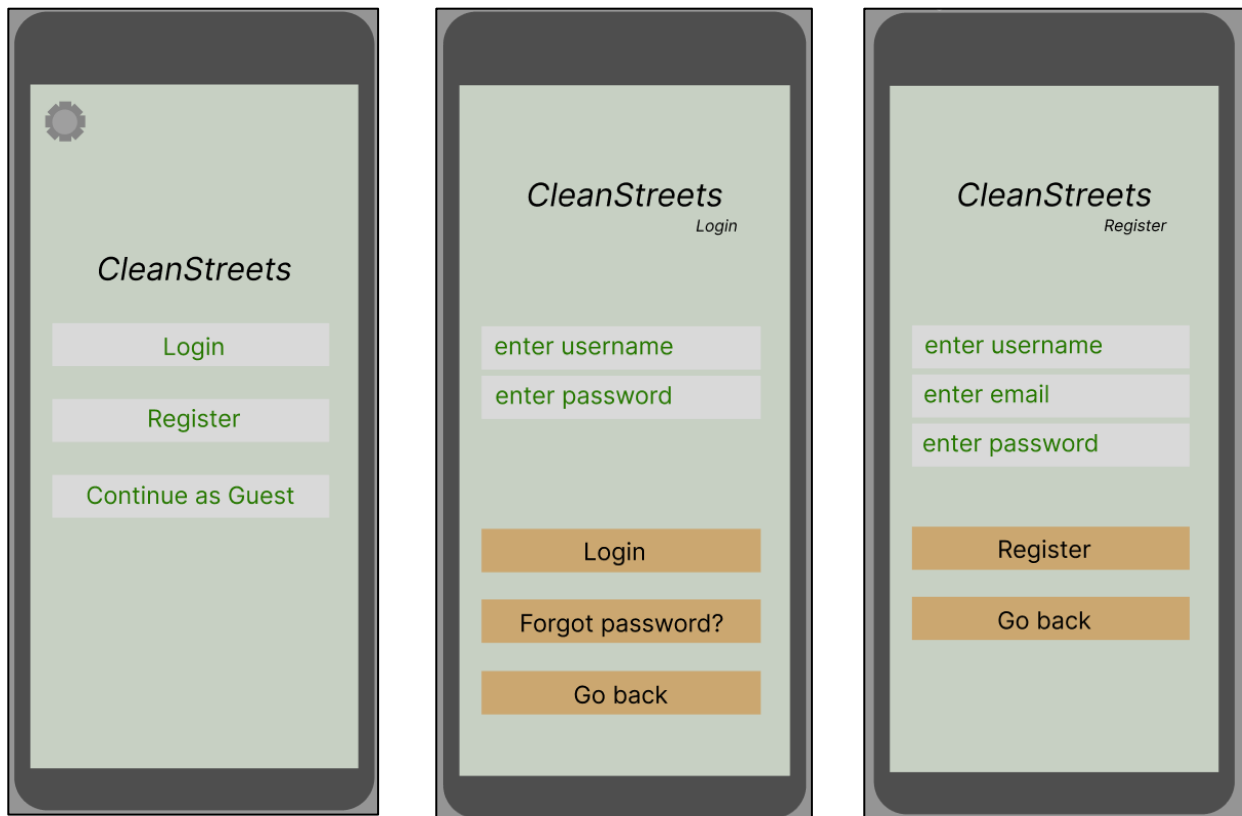


Figure 20: Medium fidelity mock-ups in Figma; Main, Login and Register screens.



Figure 21: Medium fidelity mock-ups of Navigation Bar and Reports List

Users can click a report, and they will be shown a new screen that displays a detailed view, including the image that is attached to the report.

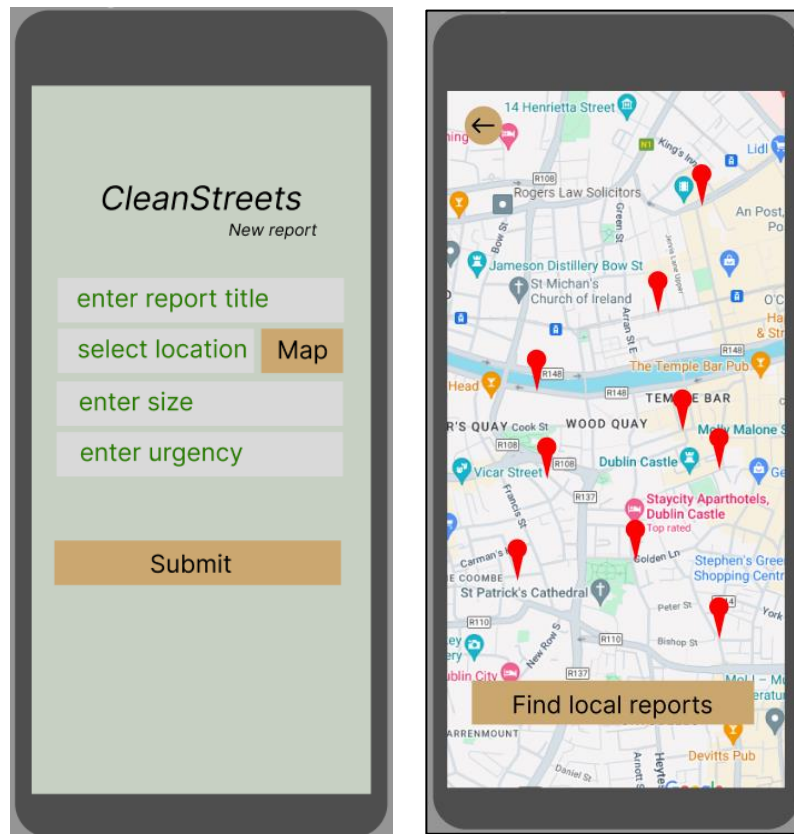


Figure 22: Medium fidelity mock-ups of New Report and Map Screens

Users can input a report title, location, and some other details such as size and an image to help others identify the problem on the street. Each pin on the map can be selected and users will be given a small description of the report. The option to find the local reports will provide a best route to the user.

3.3.3. Use Cases

Pictured in Figure 23 is the initial use case diagram (Wegmann & Genilloud, 2000). It contains three use cases that represent the account functionality, which leads into the overall core functionality of the app. The user must choose between Registering, logging in or using the app as a guest.

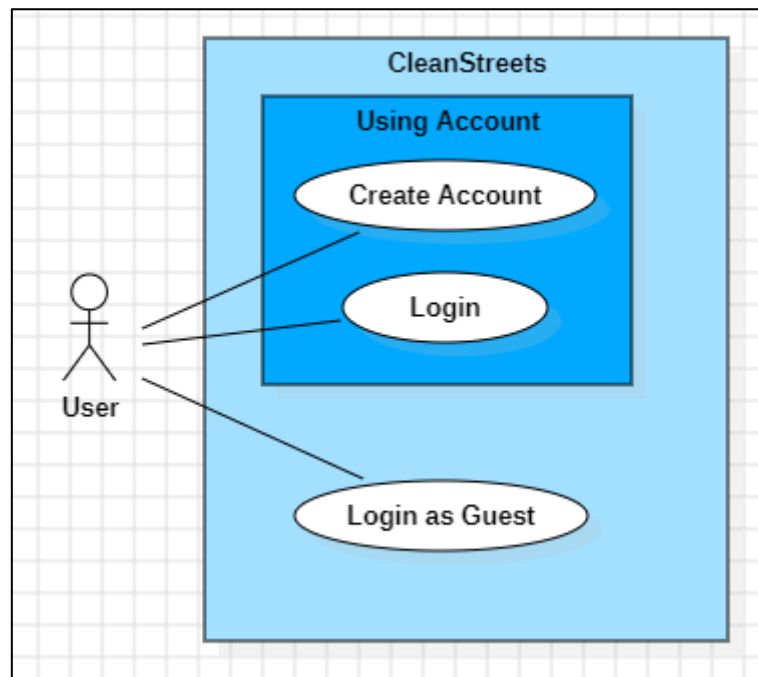


Figure 23: Initial Use Case diagram.

Depending on the user choice on the splash screen, the navigation menu that they see will be changed. For example, guest users will not see any personalised tabs such as account management, but they will be able to view a sign-up screen.

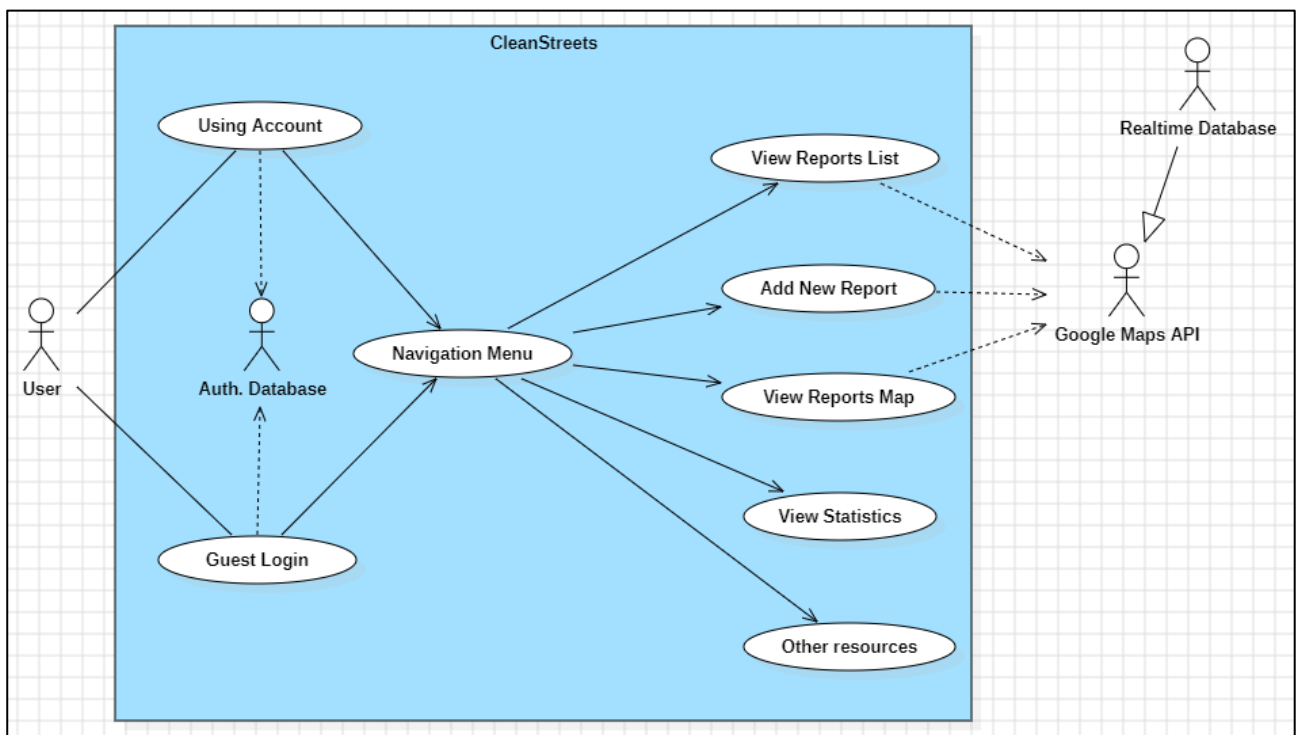


Figure 24: Extended Use Case diagram including the external connections & navigation menu.

3.4. Environment and User-Based Design Activities

3.4.1. Psychographic Exploration

As a justification and a motivation for this project, the following images were taken in one day, on a 15-minute walking journey from Arran Quay to TU Dublin Grangegorman. The tree in image 1 has pushed the slabs around it upwards, posing a danger to cyclists, pedestrians, and drivers. Unintelligible graffiti has been sprayed across a shopfront and an access gate in images 2 & 3, and five refuse sacks have been dumped in front of a doorway in image 4.



Figure 25: Examples of issues I have faced in Dublin City

Understandably, Dublin is quite a large city, and not everything can be perfect 100% of the time, however, some other examples of these quality-of-life issues can be found locally:



Figure 26: Car tyres destroyed the grass verge; Worn paint on kissing gates



Figure 27: Broken stone wall; Outdated Park paint





Figure 28: Issues faced in my locality; Overgrown bushes, defaced bin.

3.4.2. Expert Consultation

Dublin City Council was contacted regarding the possibility of their input and feedback on this project. They promptly replied, and Richard Whelan, the Administrative Officer for the Waste Management Services Division and an online meeting was organised for the following week. At the meeting.

In preparation for this meeting, a series of questions were devised to collect information about current practices in place, as well as a discussion of potential new features that this project could add to those existing practices.

The questions were varied between opinion on the project, and real-life practices, examples being:

- *What are your initial thoughts on the project?*
- *Can you tell me how reports are currently processed in Dublin City Council?*
- *Do you think that report parameters, like size and urgency, should also be pre-defined lists? (i.e., Small to Very Bulky)*
- *Do you think that the ability for users to have an account is useful in a project like this?*
- *Do you envisage this project as more community focused or more focused on helping the city council?*

- *Do you think setting a boundary around new reports is a good feature?*
- *For the reports, do you think users should be able to give a brief description of what they encountered?*

Several questions were not asked during the meeting, as some of the points covered were already discussed throughout the previous questions:

- *Do you think that the account feature would only be useful for the people who will be amending the reports?*
- *Do you think that a statistics dashboard is a feature worth having in the app accessible for ALL users?*

Some of the questions that were added were related to the existing DCC solution, explained to me during the meeting, such as how the current implementation functions, what facilities are provided to the staff and current system use among staff.

During the meeting the following key information was presented:

- The project proposal.
- The project background.
- The current state of the app.

This was followed by the interview process, where notes were kept throughout. Some key findings of this interview about the current situation includes:

- The current DCC solution, a 1-year-old web application called CitizenHub. Similar process to my app, as well as QR codes on bins.
- Mobile phone use currently in DCC; 500+ staff but phones are not provided, apart from one depot where it is on trial.

Some key discussions about potential key features of this project included:

- How a parameter such as "urgency" may not be suitable, as "my urgency may not be someone else's urgency".
- Route optimization is a good idea, as it cuts on costs and improves efficiency, DCC are working on their own algorithm for this!
- The ability to choose whether end users can have an account or to remain anonymous is well suited to the idea.
- An administrator role may prove useful, as situations like dumping are against the law and may need legal intervention. GDPR laws also need to be

respected, geocoding addresses within the app may cause unnecessary abuse to the homeowner(s) / resident(s).

- An app like this should be community-focused, but also helpful to staff when it needs to be.
- Additional ideas to consider route optimization (already planned) & image processing, i.e., blurring faces or identifying common offences.
- Pre-defined list of report options is more user friendly.
- An "in-progress" screen may be useful, as DCC receive a list of jobs to do when they start their shift, it does not update live as they go.
- Boundary is a good idea, DCC currently limits it to the DCC border.

Some additional context was also uncovered that related to the idea of this project.

- In Germany, waste collection costs are included in people's tax, and communal bins / waste areas are common. In Dublin, it is an "open market", and communal bins are rare.
- Over 3500 Bins are serviced by Dublin City Council.
- "Big Belly" bins exist, using solar energy to compact the waste, reducing collection times.
- Kerbside pickup can draw pest.
- 16,000 / 17,000 tons of waste are collected every year.
- 2 "core" depots, Northside and Southside, nearly 24/7 servicing. Outside this radius, the service reduces.
- Budget affects everything

3.4.3. Questionnaire

Additionally, a survey was conducted to explore some of the features of the system. Copies of the survey was given to classmates, as well as many international students undertaking Erasmus group, as well as some family and friends. Over 50 responses were obtained from this process, and the key questions and answers are outlined below.

1. On a scale of 1 to 5, how often do you encounter issues related to urban maintenance / waste management in your area? (*1 = Rarely, 5 = Very often*)
2. On a scale of 1 to 5, how familiar are you with existing urban maintenance / waste management systems in your city/town? (*1 = Not at all, 5 = Very familiar*)

3. On a scale of 1 to 5, how much do you believe that a mobile application can effectively contribute to improving waste management and cleanliness in your area? (*1 = Not at all, 5 = A great deal*)
4. How often do you see excessive amounts of waste or graffiti in your area?
Never, Occasionally, Frequently, Always
5. What features do you think a waste management mobile application should include? Please rank the following options based on your preference: (*1 = Low priority, 5 = High priority*)
 - a. *Reporting issues related to waste, graffiti, illegal parking etc.*
 - b. *Tracking waste collection schedules & routes*
 - c. *Providing educational resources (recycling, sustainability, etc.)*
 - d. *Providing data visualizations, insights, and trends on waste management etc.*
 - e. *Integration with local authorities' systems*
 - f. *Offering real-time updates on the status of reported issues*

➤ The first 5 questions reveal the current feeling towards the management of urban areas, as well as the groups opinion on the use of a mobile application to combat it. 59.7% of respondents indicated that they encounter issues with urban management with at least a 3/5 on the scale, with 13.5% answering 5/5 - “Very often”. Additionally, 86.5% of respondents believe that a mobile application can effectively contribute to improving waste management and cleanliness in their area.

6. Are there any specific challenges or limitations you foresee in implementing such a system? Please explain briefly.
7. Would you prefer a web-based or mobile application interface for accessing the system's community statistics, graphs and trends? Please describe briefly.
8. What are your expectations for the usability and user-friendliness of the system?

➤ The next three questions were all mandatory, forcing the respondent to give their genuine opinion on the project’s risks, implementation and design. Some of the key responses were:

- *“[may be challenging to] Create a stable communication with the local authorities.”*
- *“Needs to factor all ages. Maybe make a 15 min training video mandatory”*
- *“It should have two options for views, a simple one for an "at a glance" use case and another more detailed one.”*

- 75% of respondents prefer a mobile interface for accessing statistics & visualizations.
 - “Easy to understand and use and suitable for all levels.”
 - “Simple, intuitive and responsive.”
9. On a scale of 1 to 5, how important is scalability in such a management system? (e.g., Should it be designed to handle larger cities as well) (1 = Not important, 5 = Critically important)
 10. On a scale of 1 to 5, how likely are you to create a user account on the application? (1 = Very unlikely, 5 = Very likely, I will remain as an anonymous user)
 11. On a scale of 1 to 5, how concerned are you about the security of the data collected and stored by the system? (1 - Not concerned at all, 5 - Very concerned)
 12. On a scale of 1 to 5, how much would a convenient and user-friendly mobile application encourage you to report urban maintenance issues? (1 - Not at all, 5 - A great deal)
 13. How would you rate the urban management system in your current city or town? (Non-existent, underwhelming, average, good, excellent)
 14. Please write any additional comments here.

- The final 6 questions are about security & adoption of the system. 78.8% of respondents believe the app should be scalable to bigger cities (Minimum being a High Importance vote – 4/5). Results for creating a user account are very balanced, no clear winner or loser. Users have the choice to remain anonymous or create an account. 71.2% of respondents rate the urban management system in their current city or town as either average, underwhelming, or non-existent.

3.5. User Experience (UX) Considerations

This section explores other issues that concern the look-and-feel of system, including the screen design, and colour choices.

3.5.1. Screen design

3.5.1.1. Nielsen’s F-shape

The F-shaped scanning pattern is characterized by many fixations concentrated at the top and the left side of the page. Specifically:

1. Users first read in a horizontal movement, usually across the upper part of the content area.
2. Next, users move down the page a bit and then read across in a second horizontal movement that typically covers a shorter area.
3. Finally, users scan the content's left side in a vertical movement.

The implications of this pattern are that the:

- **First lines** of text on a page receive more gazes than subsequent lines of text on the same page.
- **First few words** on the left of each line of text receive more fixations than subsequent words on the same line. (Pernice, 2017)

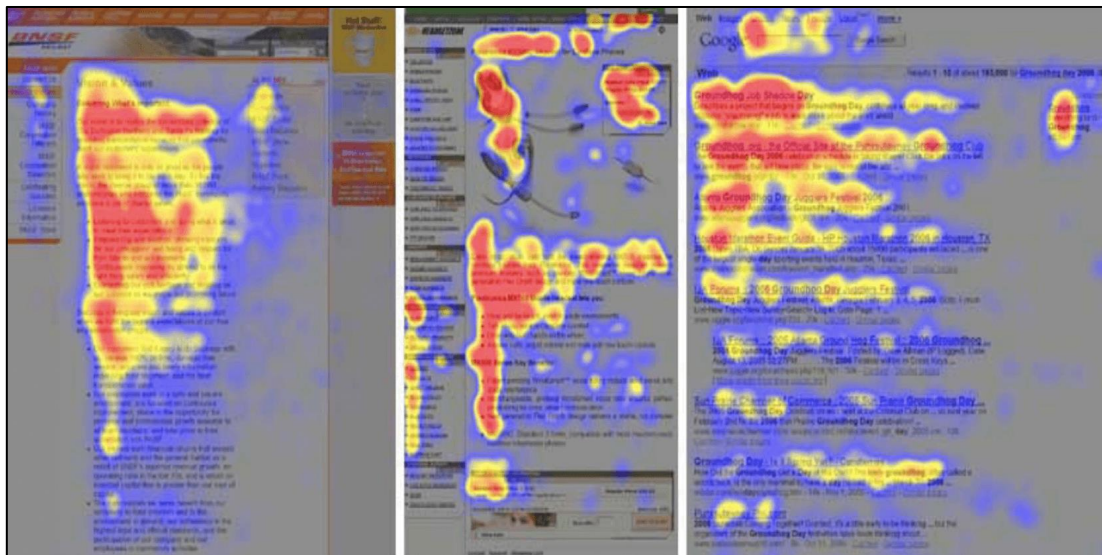


Figure 29: Heatmap of eye tracking data representing Nielsen's F-pattern (STAFF, 2019)

Since I am making a mobile application, I need to be aware of the pitfalls of Nielsen's F-pattern, and rather optimise the design of the application to ensure there is no confusion while utilizing the app, while also maintaining the structure and flow of the application.

3.5.1.2. Principles of Universal Design

There are 7 Principles of Universal Design, devised in 1997 by a working group of architects, product designers, engineers and environmental design researchers to guide the design of environments, products and communications. The seven principles are as follows: (universaldesign.ie, n.d.)

- **Equitable use.** The design is useful and marketable to people with diverse abilities.

- **Flexibility in use.** The design accommodates a wide range of individual preferences and abilities.
- **Simple and intuitive.** Use of the design is easy to understand, regardless of the user's background.
- **Perceptible information.** The design communicates necessary information effectively to the user.
- **Tolerance for error.** The design minimizes hazards and the adverse consequences of accidental or unintended actions.
- **Low physical effort.** The design can be used efficiently and comfortably and with a minimum of fatigue.
- **Size and space for approach and use.** Appropriate size and space are provided for approach, reach, manipulation, and use. (Vinney, 2021)

Each of these factors will be considered in the development of the application.

3.5.1.3. Theme

Since the application is related to keeping urban areas tidy, and ultimately helping the environment, some colour palettes were brainstormed that provide a fresh, natural feeling to the app. The website coolors.co was used for inspiration.

#6D9773	#OC3B2E	#B46617	#FFBA00

#8ECDDD	#22668D	#496800	#B48D26

#319AB9	#13647F	#2B2D42	#EDF2F4

#161A67	#993314	#933B3B	#FAF9F6

These palettes will need to be tested and evaluated like many other components of the app before being confirmed. Also, the Dublin City Council colour palette was also be considered, but ultimately, the app needs to feel playful enough for young people to use it, professional enough for adults to use it and accessible enough for elderly people to use it.

3.5.1.4. Graphic Design

There are many fonts that may suit the application, mainly Nunito, Product Sans and Montserrat. Contact was made with a Graphic Design graduate whom I met in Germany, who specialises in typography, and he offered some advice:

- **Nunito** will make it more friendly and fun because it's a rounded typeface. Should the app be friendly, and fun (like Duolingo), he suggests that Nunito should be used.
- **Product Sans** is more formal and trustworthy and more “business-y”, it's simple, clean, and straight to the point. If the app is more serious, he suggests Product Sans.
- **Montserrat** is his favourite sans serif font to use, as it is trustworthy towards the client but also retains that “playfulness”, he says. Also, Montserrat gives you more styling options. For something in between but more towards the clean and trustworthy side, he suggested Montserrat.

After evaluating his input, Montserrat will be used as my app font, pending evaluation. As the app will be used by the public, it needs to be trusted as well as easily read.

Furthermore, he designed the CleanStreets logo and app icons you can see in Figure 30. The radar-esque design of the logo subliminally tells users that they are actively “searching” for something, while also incorporating the project's initials, CS.

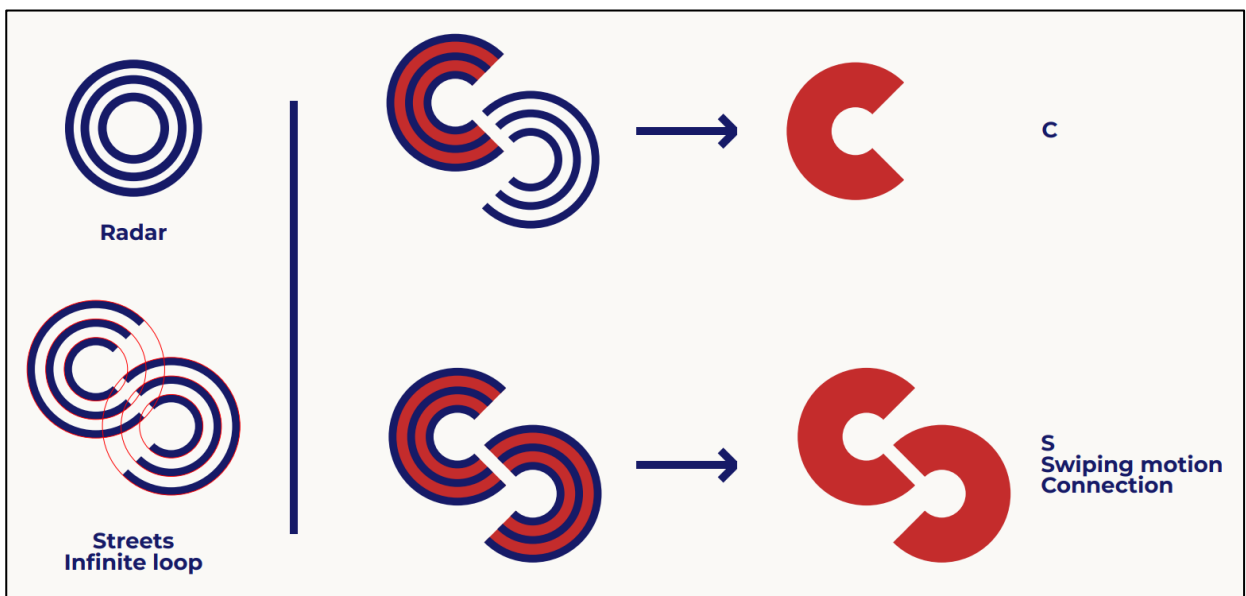




Figure 30: Logo & typography design by Neuem (Marios Andreou)

Marios also helped me choose the most suitable typography, as mentioned before, and offered some examples of how the marketing and spread of the CleanStreets project could be achieved.

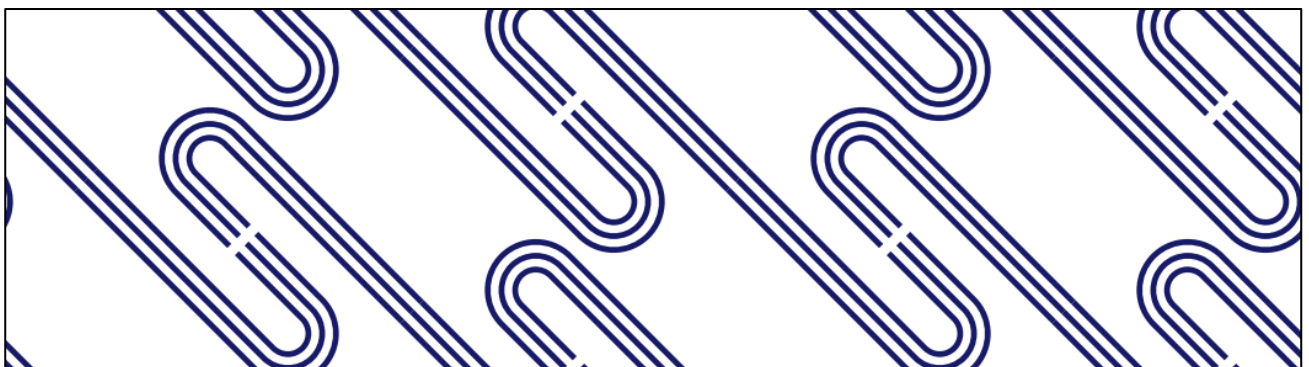
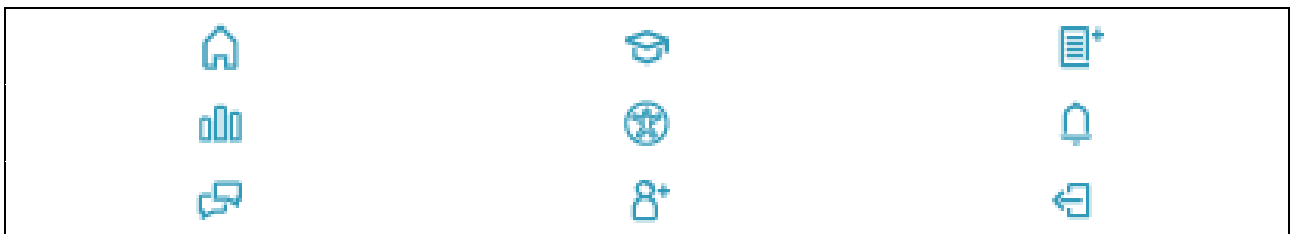




Figure 31: Typography & marketing samples (Neuem)

Finally, Neuem also designed the iconography for the CleanStreets project, a simple, clean and easily recognizable set of icons were created. These icons help to tie together the entire CleanStreets application.



3.6. Prototyping Software

Other resources that greatly helped in the design of the project include Figma, for its ease of use in creating amazing mock-ups of the app, and Dribbble, which contains thousands of existing app layouts and themes to draw inspiration from.

3.6.1. Accessibility

3.6.1.1. Language Support

The *CleanStreets* system would like to incorporate a language support accessibility feature, ensuring a user-friendly experience for a diverse audience. This feature not only caters to the needs of multicultural communities, but also enhances usability for individuals with language-specific requirements to engage with the platform effectively and contribute to the common objective.

3.6.1.2. Font Size & Colours

The development aimed to prioritize user comfort by offering adjustable text sizes as well as simplified colours, which ensures readability for users with varying visual abilities. While this was a goal at the beginning of the project, app-wide integration and support is no longer feasible given the timeframe. However, a sample screen

was created that can be expanded on in the months following the submission in order to achieve full font size support, as well as serve as an upcoming feature to existing users. (Figure 33 is altered to emulate the appearance of the simplified colours feature).

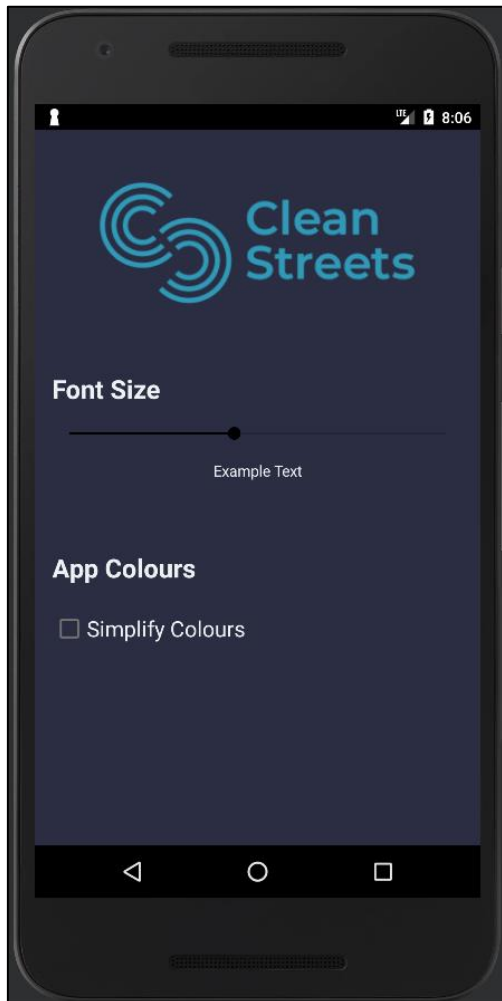


Figure 32: Sample Accessibility page

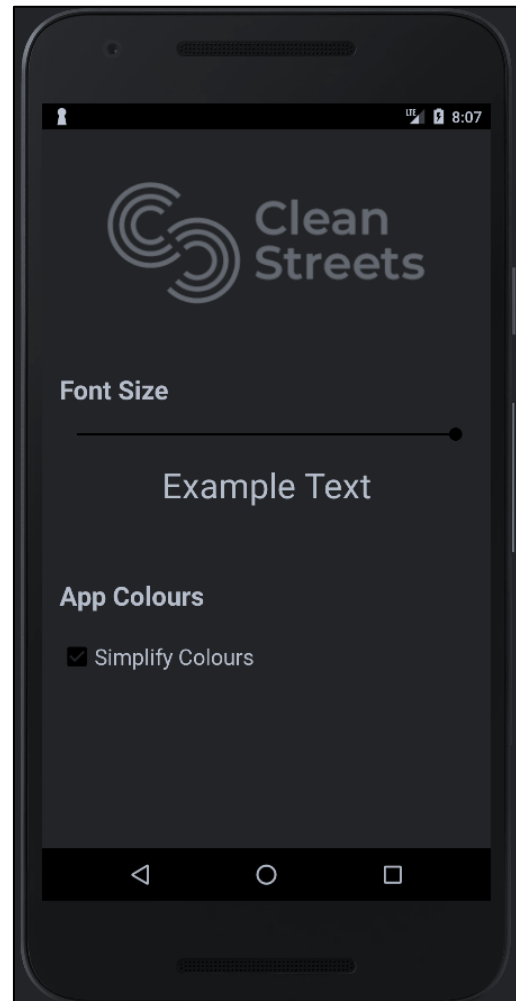


Figure 33: Emulated "Simple Colours" feature

3.6.1.3. Contrast

As mentioned in section 3.5.1.3., the app needs to be accessible and useable to many different types of people, so, after settling on a colour palette, it was evaluated against WCAG (Web Content Accessibility Guidelines) standards (WebAim: Contrast Checker, n.d.):

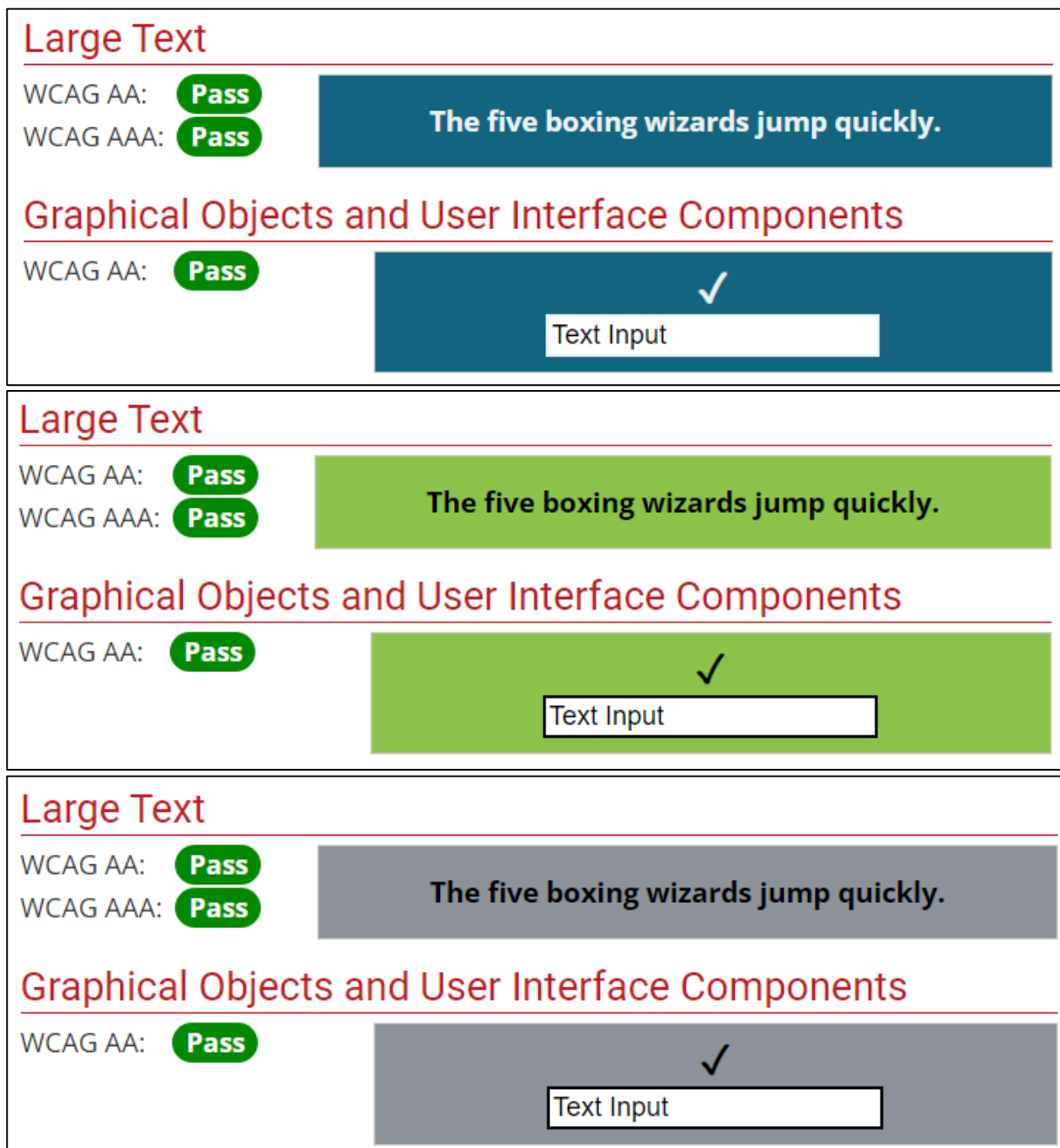


Figure 34: WCAG Contrast Checker results

3.7. Conclusions

This chapter presented an overview of the design processes that were undertaken to help develop the CleanStreets application. It begins with a discussion of the software methodology, followed by an outline of the system architecture. Following this paper prototypes and use cases are presented, and an outline of the interviews and questionnaires are discussed. Finally, some of the key User Experience (UX) considerations for this project are presented.

4. CleanStreets Development

4.1. Introduction

This chapter outlines the development of the application prototype. The planned features currently present in this prototype are the Account Management feature, Report Creation feature and the backend to frontend communication, including the map. The prototype system is based on the designs presented in Chapter 3.

4.2. Developing the Prototype

This section outlines the design decisions & considerations while developing the CleanStreets application.

4.2.1. Setting up the system

This section defines the technology stack used in the development, as mentioned in section 3.3.1.

- **Android Studio:** Primary IDE for Android app development.
- **Firebase:** For features like real-time database, authentication, and cloud functions.
- **Google Maps API:** For the mapping, location & routing services.
- **Java:** Common programming language for Android development.
- **XML:** For designing app layouts.

4.2.2. Report Creation

This section will give an insight into the report creation feature of the CleanStreets application. When the “Add Report” activity is launched, the layout is set to the contents in *activity_add_report.xml* and all the variables are initialised.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_report);

    editTextTitle = findViewById(R.id.edit_text_title); // title of the
    imageViewAttachment = findViewById(R.id.image_view_attachment); // [
    Button buttonAttachImage = findViewById(R.id.button_attach_image); //
    Button buttonSubmit = findViewById(R.id.button_submit); // submit the
    Button buttonSelectMap = findViewById(R.id.button_select_map); // for
    spinnerSize = findViewById(R.id.spinner_size); // for selecting the
    spinnerUrgency = findViewById(R.id.spinner_urgency); // for selecting
```

Figure 35: Initializing variables for reports within the CleanStreets App

Next, the Firebase database connections are initialised, and user can now add the required details into the *report* class. These details include title, size, urgency, user id, status and selected x, y coordinates from a separate map selection activity (shown in section 5.1.3.).

```
// initialize Firebase database and storage references
databaseReference = FirebaseDatabase.getInstance().getReference( path: "reports");
storageReference = FirebaseStorage.getInstance().getReference( location: "report_images");
```

Figure 36: Initializing Firebase connections within the CleanStreets App

```
.addOnSuccessListener(uri -> {
    Report report = new Report(reportId, timeStamp, selectedLatitude, selectedLongitude, size,
        urgency, uri.toString(), FirebaseAuth.getInstance().getCurrentUser().getUid(), title, status);
    saveReportToDatabase(report);
})
```

Figure 37: After error checking, the report class is created with the selected values

4.2.3. Account Management

This section will explain the methods behind the account management of the application. In the *register* activity, the user will create a username and input their email and password.

```
Ben Johnson +1
buttonRegister.setOnClickListener(new View.OnClickListener() {
    Ben Johnson +1
    @Override
    public void onClick(View v) {
        String username = editTextUsername.getText().toString().trim();
        String email = editTextEmail.getText().toString().trim();
        String password = editTextPassword.getText().toString().trim();

        // TODO: validate user inputs (check for empty fields, etc.)
    }
})
```

Figure 38: Taking user input to create an account with Firebase

Next, the Firebase built-in function “*createUserWithEmailAndPassword*” will be called with the user email and password. The new account will then be saved to the database.

```
// register the user with Firebase Authentication
firebaseAuth.createUserWithEmailAndPassword(email, password)

    .addOnCompleteListener( activity: RegistrationActivity.this, new OnCompleteListener<AuthResult>() {
```

Figure 39: Save new user to Database

The login functionality is similar, using the user input email and password to verify if the account exists in the database. Additionally, there is a third Firebase function, “*sendPasswordResetEmail*”, which unsurprisingly sends a password reset email to the user email.

```
@Override
public void onClick(View v) {
    String email = editTextEmail.getText().toString().trim();
    String password = editTextPassword.getText().toString().trim();

    //TODO: Validate the user inputs (e.g., check for empty fields)

    firebaseAuth.signInWithEmailAndPassword(email, password)
```

Figure 40: Verify user login on subsequent visits

```
@Override
public void onClick(View v) {
    String email = editTextEmail.getText().toString().trim();

    // Send password reset email
    firebaseAuth.sendPasswordResetEmail(email)
```

Figure 41: Built-in firebase password reset functionality

Users also have the choice to use the application anonymously, as the research conducted at the beginning of the development suggested users would prefer that option. Luckily, Firebase also provides an anonymous login function, so this worked exactly the same as the login feature except no credentials were required.

```
// function that allows anonymous sign-in from Firebase
private void signInAnonymously() {
    firebaseAuth.signInAnonymously()
```

Figure 42: Anonymous sign in functionality

4.2.4. Map Integration

This section will examine the functions behind the map integration within the application. In the *AndroidManifest*, the user fine and coarse location are requested. A personal Google API key is also defined here.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Figure 43: Request user-granted location permission

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyC9[REDACTED]" />
```

Figure 44: Connect to Google APIs using personal API key

Searching through a list of Report objects in *reportList*, the function retrieves the position of the input marker and compares it to the positions of reports in the list. If a report is found with a position matching the marker's position, that report is returned.

```
private Report getReportFromMarker(Marker marker) {
    LatLng markerPosition = marker.getPosition();
    for (Report report : reportList) {
        double latitude = report.getXCoordinates();
        double longitude = report.getYCoordinates();
        LatLng reportPosition = new LatLng(latitude, longitude);
        if (reportPosition.equals(markerPosition)) {
            return report;
        }
    }
    return null; // return null if no matching report is found
}
```

Figure 45: Extracting report details based off marker position

This method, *geocodeToAddress*, takes latitude and longitude coordinates as parameters and returns a human-readable address.

```
private String geocodeToAddress(double latitude, double longitude) {
    Geocoder geocoder = new Geocoder(context, Locale.getDefault());
    try {
        List<Address> addresses = geocoder.getFromLocation(latitude, longitude, maxResults: 1);
        if (addresses != null && addresses.size() > 0) {
            Address address = addresses.get(0);
            return address.getAddressLine(index: 0); // return the first line of the address
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return ""; // return an empty string if geocoding fails
}
```

Figure 46: Geocoding function, turning coordinates into human-readable addresses

Displaying detailed information about a report when a corresponding marker is selected on a map.

```
// get the Report associated with the marker and set data to TextViews
Report report = getReportFromMarker(marker);
if (report != null) {
    sizeTextView.setText("Size: " + report.getSize());
    timestampTextView.setText("Timestamp: " + report.getTimestamp());
    urgencyTextView.setText("Urgency: " + report.getUrgency());
    statusTextView.setText("Status: " + report.getStatus());
    addressTextView.setText("Location: " + geocodeToAddress(report.getXCo
}

return infoView;
```

Figure 47: Display report information from map marker

This code is responsible for placing markers on a Google Map for each Report in the reportList. Each marker contains information about the report, and when clicked, it will likely show a snippet with details like size, urgency, timestamp, status, and the geocoded address.

```

for (Report report : reportList) {
    double latitude = report.getXCoordinates();
    double longitude = report.getYCoordinates();

    LatLng location = new LatLng(latitude, longitude);
    MarkerOptions markerOptions = new MarkerOptions()
        .position(location)
        .title(report.getTitle())
        .snippet("Size: " + report.getSize() +
            "\nUrgency: " + report.getUrgency() +
            "\nTimestamp: " + report.getTimestamp() +
            "\nStatus: " + report.getStatus() +
            "\nLocation: " + geocodeToAddress(latitude, longitude));

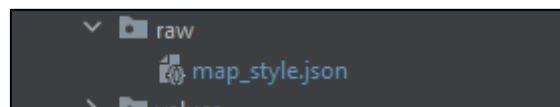
    googleMap.addMarker(markerOptions);
}

```

Figure 48: Plot the report and its details on the map

4.2.4.1. Map Style

This section will explain the choices behind the style of the map that is used within the application. The default Google Maps implementation contains a lot of detail, shops, restaurants, cinemas, the list goes on. The CleanStreets application needs to be quick and to the point, so removing all of the unnecessary, additional data points helps to speed up the backend, while maintaining the clean, accessible, interactive map.



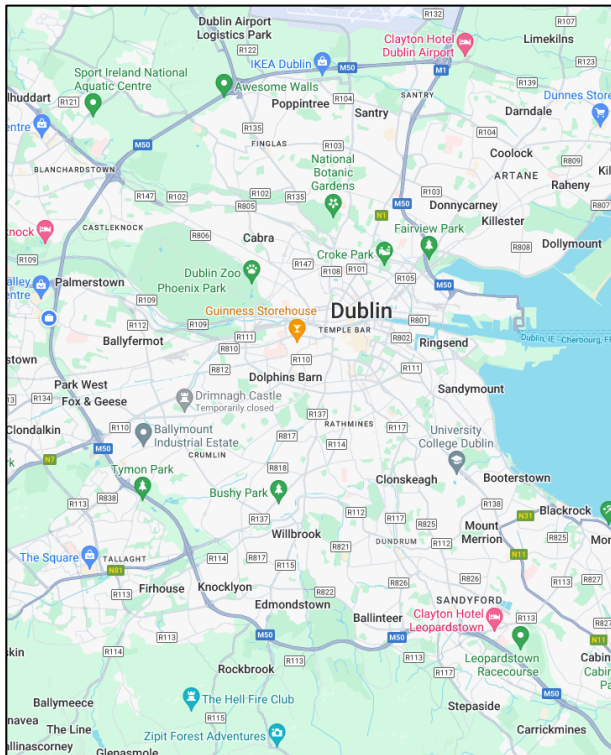


Figure 49: Default map style

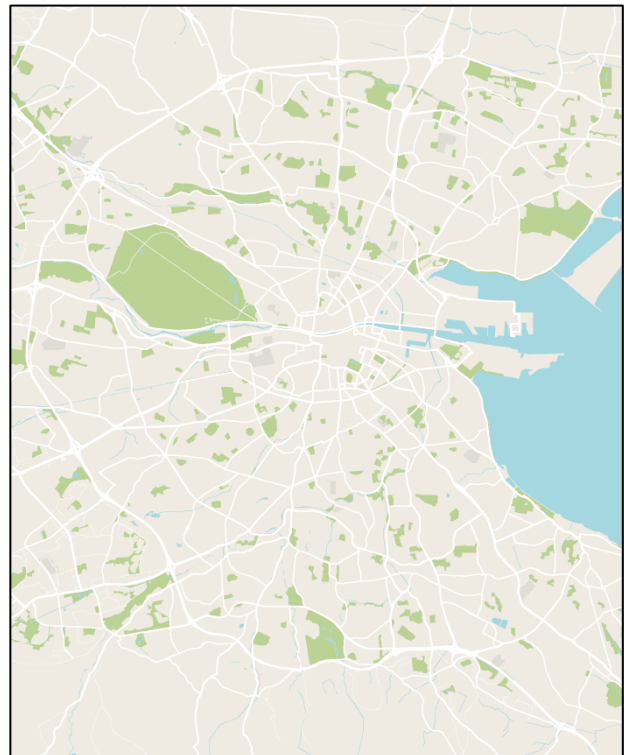


Figure 50: Production map style

4.2.5. Routing

This subsection explains the process of route finding within the application. This functionality is still incomplete at time of writing.

Function to obtain current user location.

```
private void getUserLocation() {
    // check if user has allowed location check
    if (ActivityCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
        fusedLocationClient.getLastLocation()
            .addOnSuccessListener(activity: this, location -> {
                if (location != null) { //TODO: remove current location toast

```

Figure 51: Extracting the current user location

Retrieve reports from the database, for each report, extract *LatLng* data, adds to the list, and display a corresponding toast message. If there's an issue or no coordinates are found, appropriate toast messages are displayed. Once the location of the report and user are obtained, it checks for the nearest report to the user.

```
private List<LatLng> getReportCoordinates() {
    List<LatLng> coordinates = new ArrayList<>();

    Query reportsQuery = FirebaseDatabase.getInstance().getReference( path: "reports");

    Ben Johnson
    reportsQuery.addValueEventListener(new ValueEventListener() {
        2 usages Ben Johnson
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            coordinates.clear(); // assuming coordinates is a List<LatLng>

            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                Report report = snapshot.getValue(Report.class);
            }
        }
    });
}
```

Figure 52: Populate a new list of report coordinates

```
if (report != null) { //TODO: check here for the nearest report to user
    LatLng reportLatLng = new LatLng(report.getXCoordinates(), report.getYCoordinates());
    Toast.makeText( context: RouteFinderActivity.this, text: "Report Location: " + reportLatLng,
        coordinates.add(reportLatLng);
}
```

Figure 53: Using the list of coordinates to find the nearest report to the user

Once this function is completed, an API call is made to Google Maps, which finds the best route between the two points known as a “polyline” and placed on the map. The polyline needs to be parsed from the JSON response.

```
// URL for Google Directions API
String url = "https://maps.googleapis.com/maps/api/directions/json?origin="+extractCoordinates(String.valueOf(origin))+"&destination="+
    extractCoordinates(String.valueOf(destination))+"&mode=walking&key="+apiKey;
```

Figure 54: Google Maps API call

```
private static String extractPolyline(String jsonResponse) {
    // Parse the JSON response
    JsonParser jsonParser = new JsonParser();
    JsonObject jsonObject = jsonParser.parse(jsonResponse).getAsJsonObject();

    // Check if there are routes in the response
    if (jsonObject.has( memberName: "routes") && jsonObject.getAsJsonArray( memberName: "routes").size() > 0) {
        // Extract the polyline string from the "points" property of "overview_polyline"
        JsonObject route = jsonObject.getAsJsonArray( memberName: "routes").get(0).getAsJsonObject();
        if (route.has( memberName: "overview_polyline")) {
            JsonObject overviewPolyline = route.getAsJsonObject( memberName: "overview_polyline");
            if (overviewPolyline.has( memberName: "points")) {
                return overviewPolyline.get("points").getString();
            }
        }
    }
}
```

Figure 55: Extracting polyline from JSON response

4.2.6. Noticeboard

This part of the report will outline the functionality of the 'Noticeboard' feature. Paul Bourke, a lecturer at TU Dublin, who is also overseeing this project, suggested a "community" focused feature that was not identified at the beginning of the development. This feature was added after evaluating his input.

Users of the CleanStreets application, who have an authenticated account, will be able to create and reply to posts regarding events and discussions happening in their area. Firstly, users will add details like the title, content, postcode and tag:

```
private void submitPost() throws IOException {  
    // initialise post parameters  
    final String title = postTitle.getText().toString().trim();  
    final String content = postContent.getText().toString().trim();  
    final String postcode = spinnerPostcode.getSelectedItem().toString();  
    final String tag = spinnerTag.getSelectedItem().toString();  
}
```

Figure 56: Uploading a 'post' snippet

Similarly to the report upload, the post is sent to the Firebase database:

```
Post post = new Post(FirebaseAuth.getInstance().getCurrentUser().getUid(), referencedReportId: null, postId,  
    timeStamp, title, content, tag, postcode);  
  
databaseReference.child(postId).setValue(post)  
    .addOnSuccessListener(new OnSuccessListener<Void>() {  
        @Override  
        public void onSuccess(Void aVoid) {  
            Toast.makeText(context, AddPostActivity.this, text: "Post added successfully!", Toast.LENGTH_SHORT).show();  
            finish();  
        }  
    })  
    .addOnFailureListener(new OnFailureListener() {  
        @Override  
        public void onFailure(@NonNull Exception e) {  
            Toast.makeText(context, AddPostActivity.this, text: "Failed to add post: " + e.getMessage(), Toast.LENGTH_SHORT).show();  
        }  
    });
```

Figure 57: Sending 'post' to Firebase

As users need to have an account in order to use this feature, when they reply to anything, their unique ID is attached to every reply, as well as the ID of the report they are replying to:


```
// current user's ID
String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();

// new reply object
Reply reply = new Reply(userId, replyMessage);

// Push reply to firebase
repliesRef.push().setValue(reply);

Toast.makeText( context: this, text: "Reply sent successfully", Toast.LENGTH_SHORT).show();
```

Figure 58: Save user uid and 'reply'



Figure 59: Example screenshot of replies

4.2.7. Word filtering

This is a small part of the report dedicated to explaining the word filtering and censoring involved in the CleanStreets project. A '*BadWordsFilter*' was developed that scans all of the relevant user input strings. Excluding the check on email input was decided upon as although it may seem rude or inappropriate, some people's email addresses will contain obscenities. A text file (*bad-words.txt*, (Carnegie Mellon University, n.d.)) was sourced online and this is what the class bases it's filtering on.

```

public boolean containsSwearWord(String input) {
    String[] words = input.split( regex: "\\s+" );
    for (String word : words) {
        if (badWords.contains(word.toLowerCase())) {
            return true;
        }
    }
    return false;
}

```

Figure 60: Snippet of code showing the containsSwearWord function

```

// bad words check
BadWordsFilter badWordsFilter = new BadWordsFilter(getResources().getAssets().open( fileName: "bad-words.txt" ));

if (badWordsFilter.containsSwearWord(title)) {
    Toast.makeText( context: this, text: "Do not use inappropriate language, try again.", Toast.LENGTH_SHORT).show();
    return;
}

```

Figure 61: Instantiation & usage of BadWordsFilter

4.2.8. Analytics

This section will examine the “analytics” feature, where users can view various statistics about the data currently contained in the database. At present, there are 5 different charts, representing Report & Post variables like Size, Status and Tags. The MPAndroidChart library was used to generate the charts.

```

Report report = snapshot.getValue(Report.class);
if (report != null) {
    String status = report.getStatus();
    switch (status) {
        case "Active":
            activeReports++;
            break;
        case "In-progress":
            inProgressReports++;
            break;
        case "Completed":
            completedReports++;
            break;
    }
}

displayPieChart(chart4, activeReports, inProgressReports, completedReports);
}

```

Figure 62: Pre-processing of the data for analytics feature

The processing is quite simple for all charts, retrieve the data from Firebase, use counters for each relevant variable, display the data on each chart.

4.2.9. Administrative Role

This section will discuss the role of “admin” users, and the responsibilities that they have throughout the application. An “isAdmin” variable is assigned to all users (default is “No”), before performing administrative tasks, the program will check for this variable to be equal to “Yes”, otherwise the task is not available.



Figure 63: isAdmin variable in user database

```
usersRef.child(currentUserId).addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        // check if the current user isAdmin
        if (dataSnapshot.exists() && dataSnapshot.child("isAdmin").getValue(String.class).equals("Yes")) {
            // show amend button
            amendButton.setVisibility(View.VISIBLE);
        } else {
            // hide amend button
            amendButton.setVisibility(View.GONE);
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
        // Handle error
        Toast.makeText(context, DetailedReportActivity.this, "Failed to load user data.", Toast.LENGTH_SHORT).show();
    }
});
```

Figure 64: Admin only amend report option

4.2.10. Report upvotes

This section will briefly describe the thought process and implementation of the report upvote system available to all users, on all reports. It is quite simple, an `onClick` function was set up using the thumbs-up icon, calling the “`incrementThumbsUpCount()`” function, which then refreshes the UI.

```
thumbsUpIcon.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        int position = getAdapterPosition();  
        if (position != RecyclerView.NO_POSITION) {  
            // get report at this position  
            Report clickedReport = reportList.get(position);  
            // increment the thumbs up count for this report  
            clickedReport.incrementThumbsUpCount();  
            // update the UI  
            updateThumbsUpCountLocally(clickedReport.getThumbsUpCount());  
        }  
    }  
});
```

Figure 65: thumbsUpIcon OnClick Listener

```
public int getThumbsUpCount() {return thumbsUpCount;}  
  
public void setThumbsUpCount(int thumbsUpCount) {this.thumbsUpCount = thumbsUpCount;}  
  
public void incrementThumbsUpCount() {thumbsUpCount++;}
```

Figure 66: Report class thumbs-up getters and setters

The upvote feature was added to the app as it seemed viable, if you also seen something to report, and somebody already reported it, you simply add a like to their report. However, this project is not social media, and considerations had to be made for the dangers of adding features like “upvotes” and in section 4.2.11, replies. “*These apps, through likes, comments, and shares, offer instant gratification, potentially fuelling addictive tendencies*”, (Jo & Baek, 2023).

4.2.11. Post replies

This section will highlight the process of the reply feature, available to all authenticated users on all noticeboard posts. When users click the “Add Reply” button, they are redirected to a new activity where they can see other replies & write their own reply, and subsequently send it.

```
// get postId from intent
String postId = getIntent().getStringExtra( name: "post_id");

// firebase reference for replies
repliesRef = FirebaseDatabase.getInstance().getReference( path: "replies").child(postId);
```

Figure 67: Retrieve post_id and load existing replies

```
// error checking
if (message.isEmpty()) {
    Toast.makeText( context: this, text: "Please enter a reply message", Toast.LENGTH_SHORT).show();
    return;
}

// bad words check
BadWordsFilter badWordsFilter = new BadWordsFilter(getResources().getAssets().open( fileName: "bad-words.txt"));

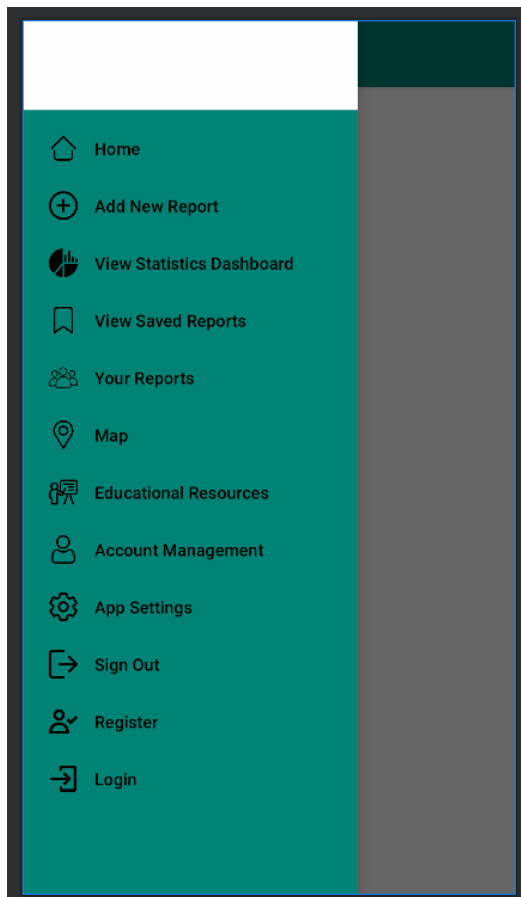
if (badWordsFilter.containsSwearWord(message)) {
    Toast.makeText( context: this, text: "Do not use inappropriate language, try again.", Toast.LENGTH_SHORT).show();
    return;
}
```

Figure 68: Reply error / swear checking

The replies screen also contains some basic error checking and an instance of the bad words filter to prevent any hurtful messages and protect the user.

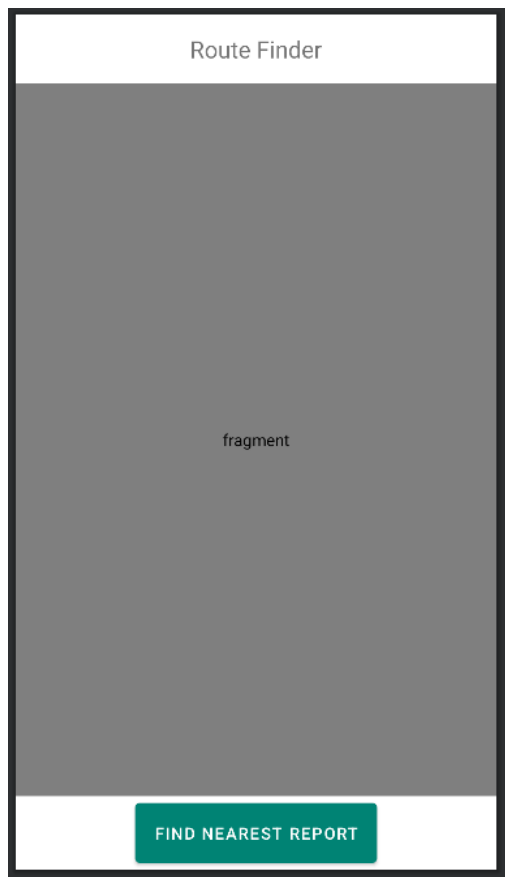
4.3. System Running

Below are some images of the system in action:



The image shows a mobile application's register screen. It has a white background with a dark teal header. The form includes three input fields: Username, Email, and Password. Below the input fields are two buttons: a teal button labeled 'REGISTER' and a teal button labeled 'HOME'.

Figure 69: Early development navbar and register screens



The image shows a mobile application's report creation screen. It has a white background with a dark teal header. The form includes five input fields: Report Title, Enter Report Title, Location, Size (with a dropdown arrow), Urgency (with a dropdown arrow), and Attachment. Below the input fields are two buttons: a teal button labeled 'ATTACH IMAGE' and a teal button labeled 'SUBMIT REPORT'.

Figure 70: Early development route-finding and report creation screens

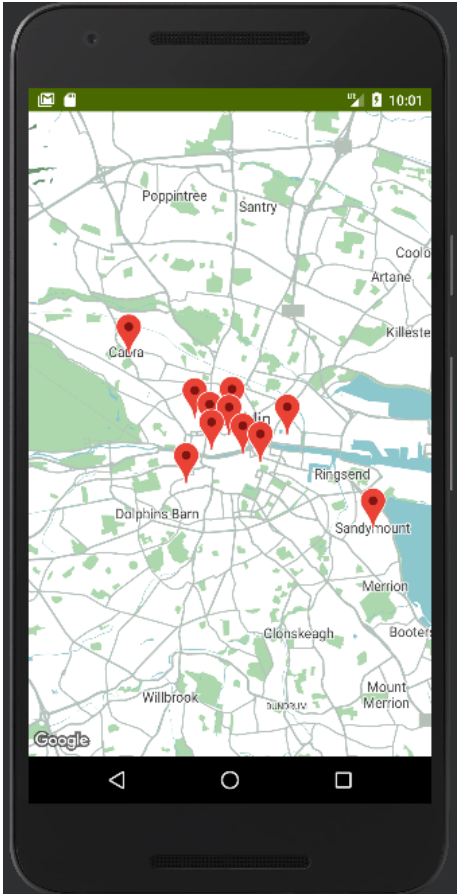


Figure 71: Mid-development splash & reports map screens

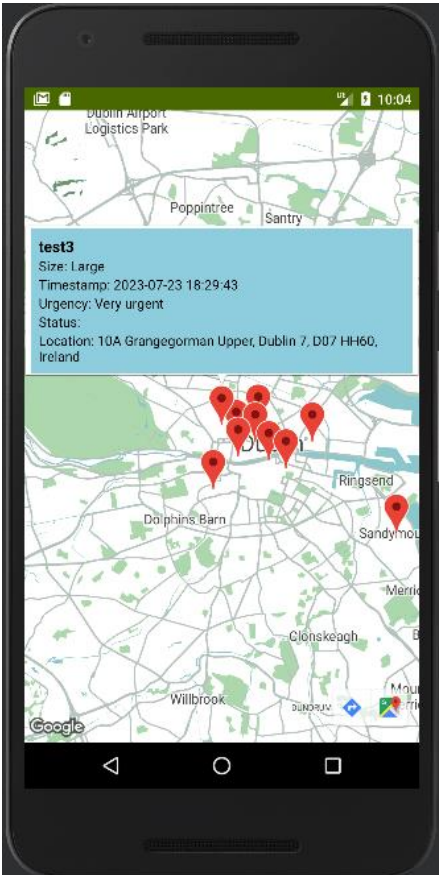
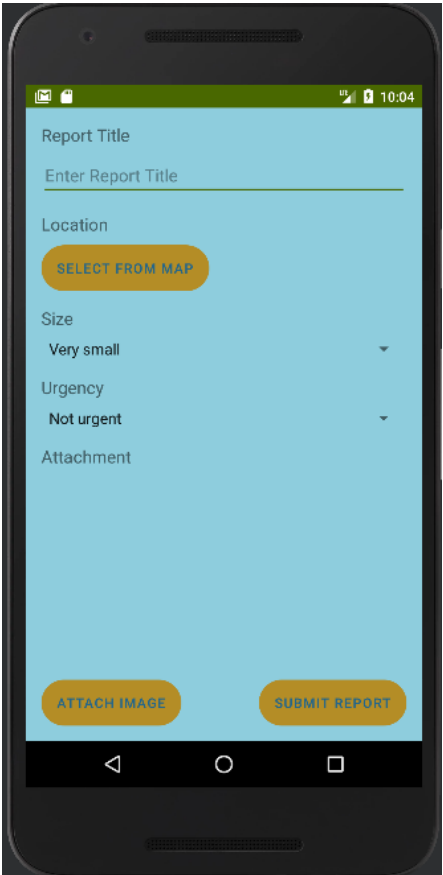


Figure 72: Mid-development report creation & detailed report view screens

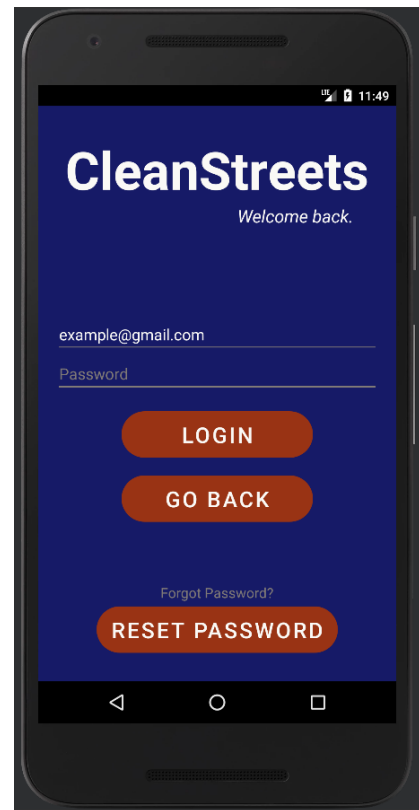


Figure 73: Late development main screen & Login

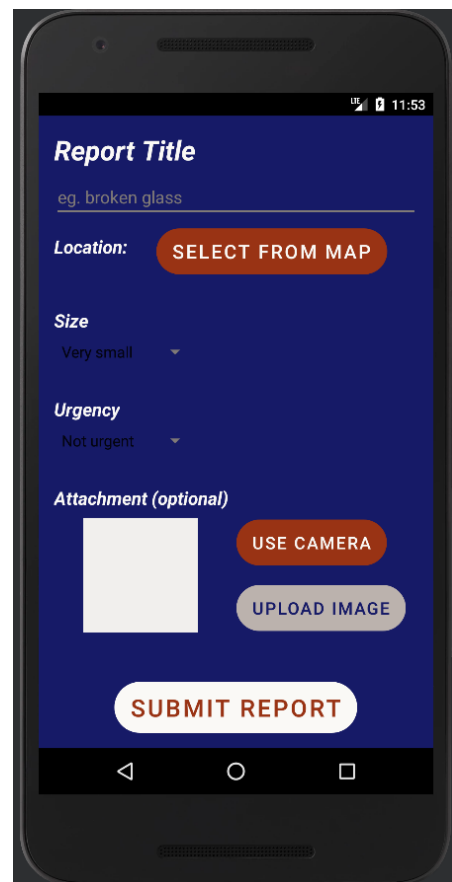
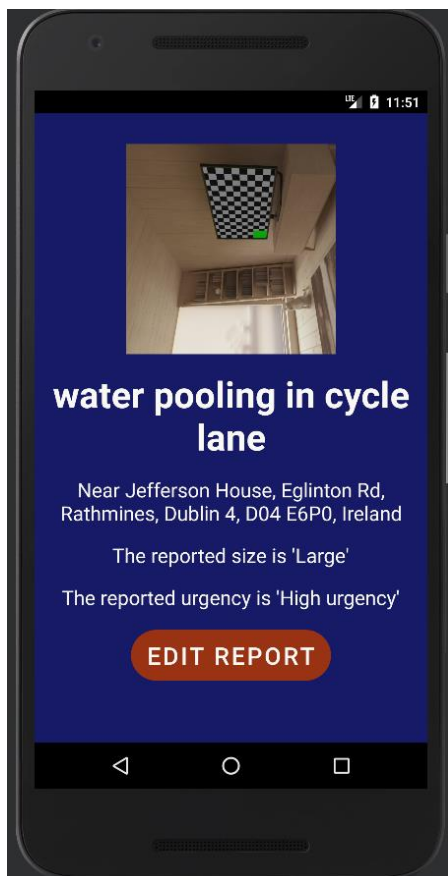
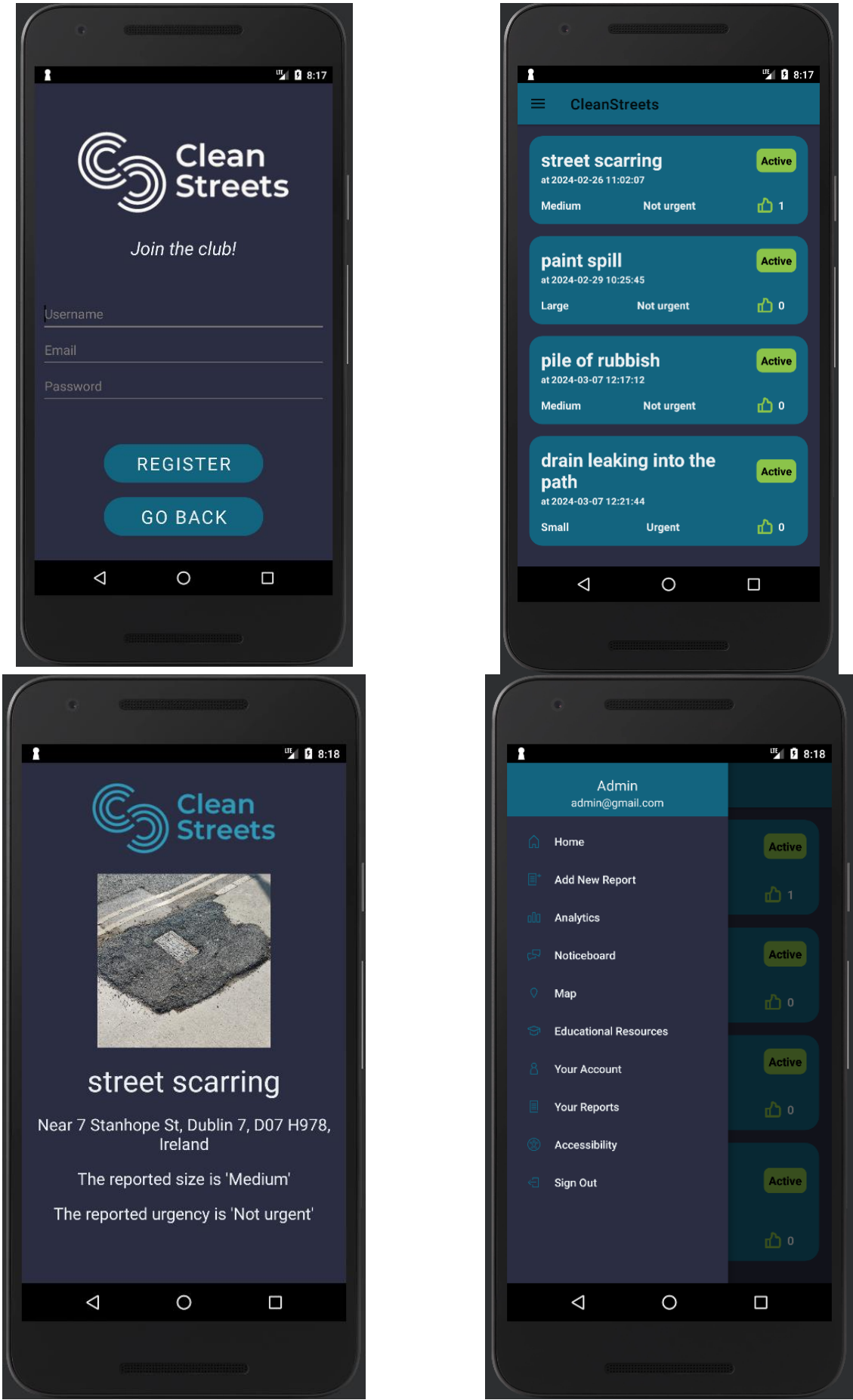


Figure 74: Late development detailed report and add report screens

The following images are the final product, as they appear on user devices.



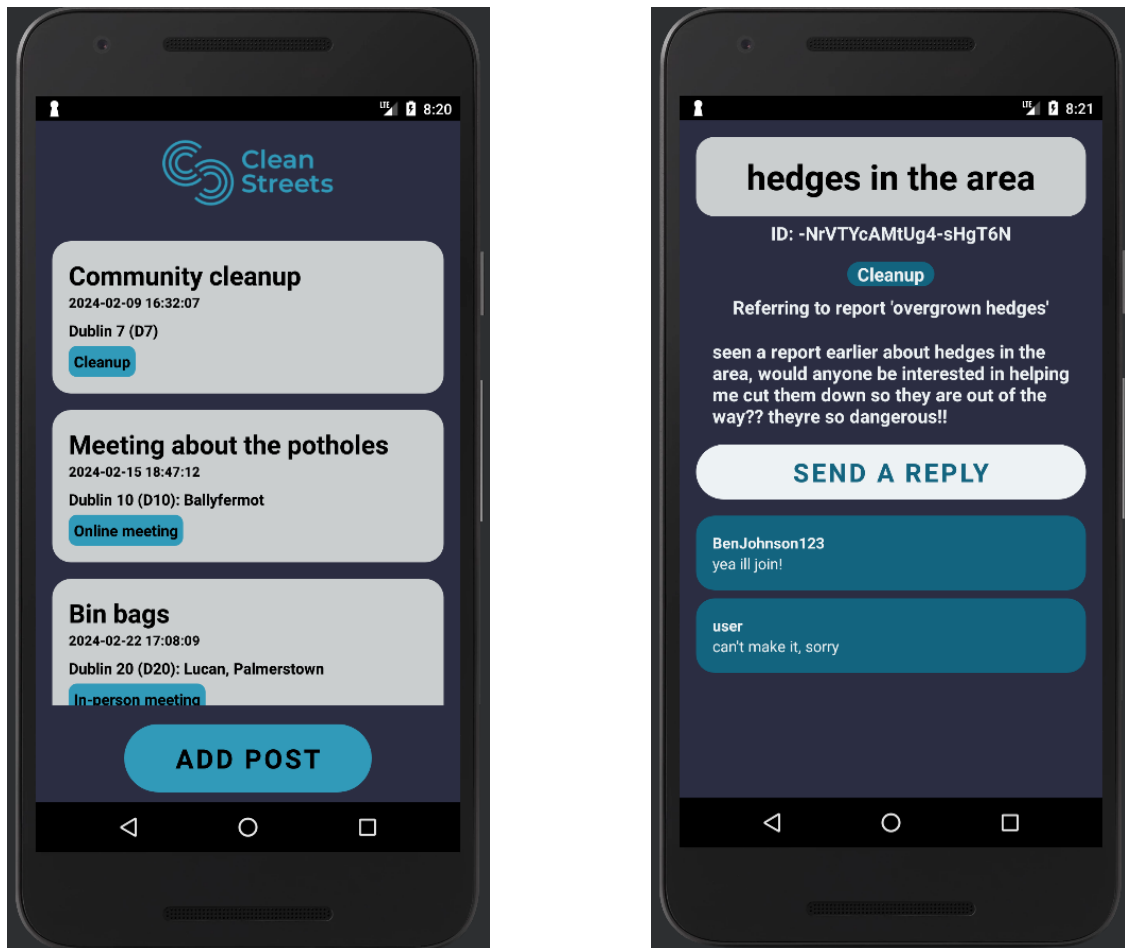


Figure 75: Final product screenshots

4.4. Key Development Challenges

It is worth highlighting some of the key development challenges in the creation of the CleanStreets Application, that includes simulator challenges, configuration challenges, and programming challenges.

- Emulator “location” was locked to the server location (San Diego, California). This made route finding impossible to test until a physical device was used. This issue was handled by hard coding the location to around 53, -6 (Centre of Dublin City).
- Font size, weights, types could not be truly tested until the application was put onto a physical device. Using default, pre-set fonts was tough as it was hard to understand how the app would turn out in the end. Thankfully, the graphic designer knew what he was doing.
- Parsing the polyline response from the Google Maps API call was a challenge. It is buried so deep within a JSON response each time. It was overcome by designing modular loops to extract only the polyline each time, for the nearest report based on user location.

- The use of the camera was a significant problem, as it is one of the core features of the project. The emulator always generates the same image, not considering blur, angles, zoom, and often would flip the image 90°. Again, the solution was a well-designed algorithm and testing on a real phone.
- At the beginning of development, it was difficult to separate users with an account and those without. It took time to understand how Firebase Accounts work, including instantiating user objects for verification in certain feature screens.
- Allowing users to change their username was a challenge as Firebase only comes as Email and Password accounts. An additional Realtime Storage table was set up, linked to each users unique ID in order to track their username, and changes, correctly.

4.5. Conclusions

In this chapter, the development process for the prototype was identified. Account management, report creation and map integration were discussed. With the prototype developed and testing & evaluation examined, we can move on to related issues and the future potential of the CleanStreets application in the final chapter.

5. Testing and Evaluation

5.1. Introduction

This chapter will highlight the processes and methods used to test and evaluate the project. The testing process includes the development of test plans and unit testing, and the evaluation process will look at both UX and user-centred techniques to assess the quality of the outcomes of the project.

5.2. Plan for Testing

The *CleanStreets* application will be tested at all tiers (front, middle, and back) using a variety of methods. Manually testing for bugs, errors, and other anomalies will be used to test specific features and one-off scenarios. Automated tests may also be developed to test use-cases from the end-user's perspective to evaluate the application's usability, functionality, and performance. (Unadkat, 2023)

A Unit Testing plan will be created and carried out later in the development. There will also be considerations for black, grey and white box testing; *“black box testing is often used for validation (i.e., are we building the right software?) and white box testing is often used for verification (i.e., are we building the software right?)”* (Nidhra, 2012)

A sample plan is presented below:

#	Description	Pass	Fail
1	Does the CleanStreets icon appear on my phone screen?	✓	
2	If I click on the icon, does it open the CleanStreets system?	✓	
3	When the system opens, does it give me a splash screen?	✓	
4	After the splash screen, do I get a menu page with options?	✓	
5	Does the system load the user logon screen when I select that option?	✓	
6	Does the system load the user logon settings when I select that option?	✓	
7	Does the system run the camera function when I select that option?	✓	
8	Does the system exit when I select that option?	✓	

5.2.1. Local Testing

This section will highlight the methods used to test the CleanStreets project locally.

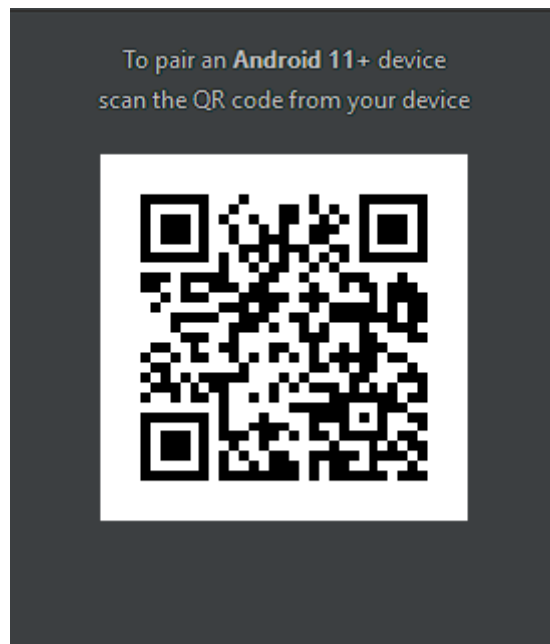
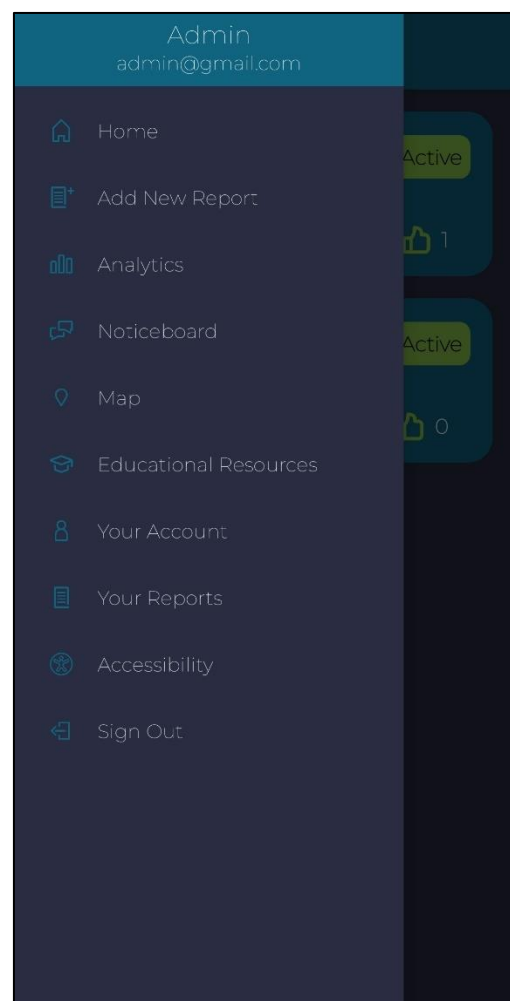
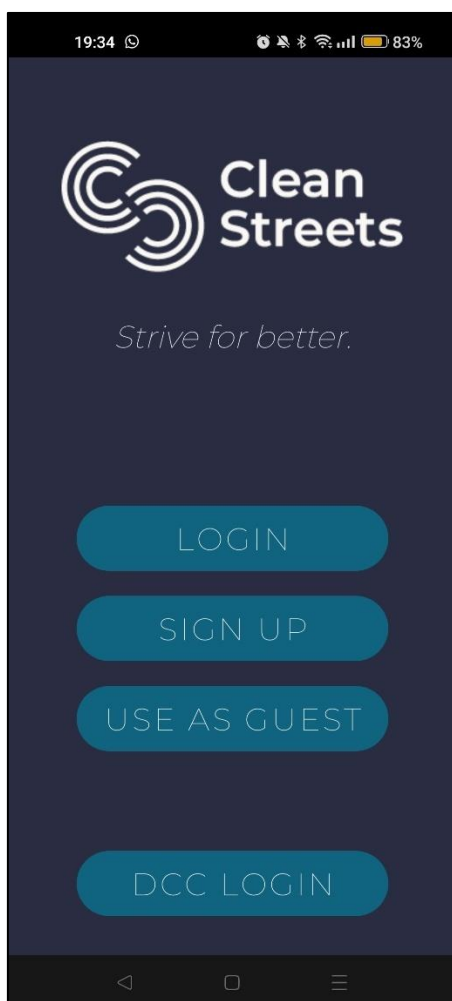


Figure 76: Android Studio debugging QR code

As this project is an Android-focused development, Android Studio's debugging tools were used, including the QR code which enables debugging and testing on physical devices. The images included below are screenshots from a physical device.



5.3. Plan for Evaluation

This section will briefly highlight the methods that will be used to evaluate the CleanStreets project, including user evaluation, expert evaluation, and automated evaluation. These users will be asked to use the system, and then perform several specific tasks on the system which will be monitored and following this a combination of interviews and surveys will be used to identify any key challenges in the system. (Cresswell & Miller, 2000)

5.3.1. User Evaluation

This subsection explains several user-focused evaluation methods that will be used to evaluate the CleanStreets app. Following Jakob Nielsen's advice which states that *"using five test participants in a usability test will discover 85% of the usability problems that there are to find"* (Nielsen, 2000), I can safely evaluate the usability of the application with a minimum of 5 testers.

5.3.1.1. Students Learning with Communities TUD

TU Dublin offers a community-engaged learning and research initiative, where staff, students and community partners collaborate to develop real-life projects for mutual benefit. This project will make use of the opportunity.

5.3.1.2. Non-Expert Evaluation

To get a real sense of the unpredictable nature of developing and deploying an application, the CleanStreets project will be shown to members of my family and friend groups to gather their thoughts, feelings, issues and general feedback. Testers were asked to briefly describe a few things that they like about the application, some things they dislike and some things that they would change or add. The following results are direct quotes from the testers:

I like that it is ...	I would like ...
✓ Easy to use ✓ UI is intuitive ✓ Colours are very nice ✓ Reporting is easy ✓ Account management is easy ✓ I like seeing all of the reports in one place ✓ Analytics page is nice to have ✓ Nice font ✓ Functions smoothly, no lag ✓ Noticeboard feature is good	+ I would like a confirmation box when selecting a report location from the map + I would like to see the navbar present on all screens + Map labels (for streets & roads) + Adding multiple images to a report + Prompt to use current location for a report + Email confirmation of report status / completion

And two key challenges noted were:

- Loading issues (2)
- Formatting of the analytics labels is bad

One tester enquired about the security of the database in relation to SQL injection, but it was explained to them that Firebase utilises API calls instead of direct SQL commands, so there is no risk of SQL injection.

The loading issues identified are down to the weak internet connection in the testing location, other testers do not identify loading issues once the testing location has good internet connection.

Additionally, more offensive and derogatory words were identified and subsequently added to the bad words filter.

5.3.2. Expert Evaluation

This subsection explains several expert-specific evaluation methods that will be used to evaluate the CleanStreets app.

5.3.2.1. Dublin City Council

At the conclusion of the meeting with Dublin City Council, Richard said he would be more than happy to organise another meeting near the end of the project for some final feedback and evaluation.

5.3.2.2. Accessibility Experts

Two Accessibility Experts reviewed the system to evaluate it for any challenges it may present to people with visual impairments and other disabilities. One of the two experts works for the National Disability Authority (NDA) and has over a decade of experience undertaking accessibility reviews for both webpages and apps, and the other has over 20 years' experience reviewing a range of software applications for accessibility challenges and has worked with the National Disability Authority (NDA) and the National Council for the Blind of Ireland (NCBI). The feedback from the experts was as follows:

(In terms of the thickness of the white text in the buttons, can you make them thicker, just for the demo video? or do a higher resolution video?)

- **Menu screen** - Clear layout, with large, clear buttons. The white text is very thin in the buttons.
- **Sign-in screen** - Clear layout, with large, clear buttons. The white text is very thin in the buttons.
- **Login screen** - Clear layout, with large, clear buttons. The white text is very thin in the buttons.

- **Home screen** - Could you put the word "REPORTS" at the top of this page? List of reports - very clear, **nice colour contrast** between blue and gold/green "Active" status, and "Like" button. I know the Menu button is a hamburger button, but could you put the word "MENU" beside it as well, to be sure people will get it?
- **Report screen** - We only see it for a second at 1:07, but can you add a heading above the photo "REPORT #121712" etc.
- **Add report screen** - Really simple and **clear usability with very good labelling**.
- **Analytics screen** - Great colours, excellent diagrams, very clear layout, **will work for people with Colour Blindness**, and other visual impairments.
- **Map screen** - Very clear layout
- **Accessibility Screen** - I like it!!! Does the Simplify Colours work?
- **Admin Login screen** - Very Clear
- **Noticeboard screen** - Could you put the word "NOTICEBOARD" at the top of this page? And maybe a logo something like this?



Figure 77: Accessibility expert recommendation

- **Account screen** - Very clear

The accessibility experts were very happy with the font choice, colour and size. One expert noted that it achieved the goal set in Section "3.5.1.3. Theme", of being playful yet accessible to everyone.

5.3.3. Automated Evaluation

Test methods will be created for some of the main functions present in the CleanStreets application. These test methods will then be run under various conditions to evaluate the effect on the system, its subsequent functionality and security.

A repository evaluation tool, Snyk (Snyk, n.d.), was used to analyse and identify issues and vulnerabilities within the CleanStreets project. The results were as follows:



Figure 78: Summary of security issues from Snyk

3 High risk, 8 Medium risk and 3 Low risk issues were found. One low-risk issue was related to “JavaScript Enabled”, which if used maliciously, *“may leave the application exposed to attacks like Cross-Site Scripting, Information leakage or Remote Code Execution if untrusted sources are loaded.”*

The other two low-risk issues were related to information exposure – *“Affected versions of this package are vulnerable to Information Exposure. When there is no byte array value that can be encoded into a string the Base32 implementation does not reject it, and instead decodes it into an arbitrary value which can be re-encoded again using the same implementation. This allows for information exposure exploits such as tunnelling additional information via seemingly valid base 32 strings.”*

The Medium-risk issues involved some of the packages that were used to help the app function, not necessarily the code that had been written:



Figure 79: Three examples of project vulnerabilities

Finally, the high-risk issues were related to Denial-of-Service just like the previous issues, however there was one that was about *“Allocation of Resources Without*

Limits or Throttling”, that was unfortunately introduced alongside the AndroidMPChart library.

Detailed paths

- Introduced through: `com.github.mikephil:MPAndroidChart@1.4.2-SNAPSHOT` > `com.google.android:android@4.1.1.4` > `org.json:json@20080701`
Fix: No remediation path available.

Security information

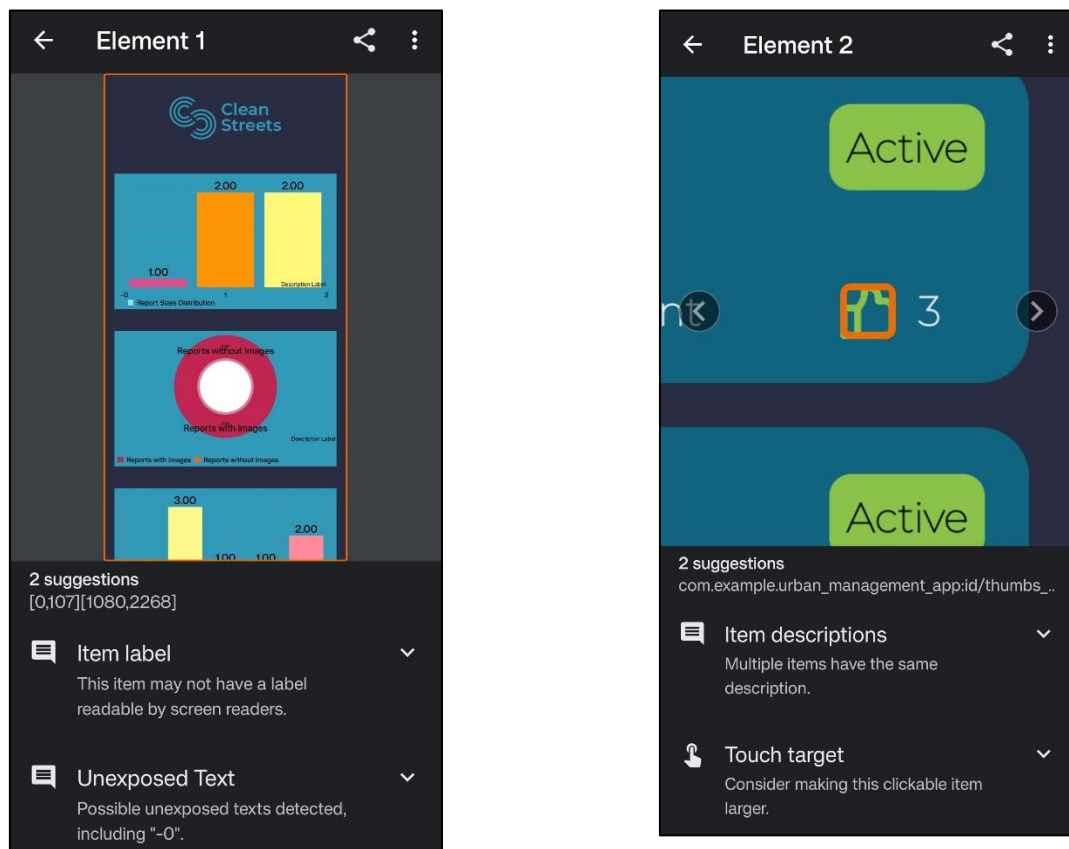
Factors contributing to the scoring:

- Snyk: CVSS 7.5 - High Severity
- NVD: CVSS 7.5 - High Severity

Figure 80: Allocation of resources high-risk issue

Some of these issues have explanations, for example, “Deserialization of untrusted data” is not a problem as I am using a private Google Maps key to retrieve the polyline from an API call, mentioned in Section 4.2.5., and the issue related to information exposure can be exploited by using a large string input, but I have limited the strings for user input to maximum 60 characters.

Additionally, I used Google’s Accessibility Scanner on a physical device to identify accessibility problems, and then amend them.



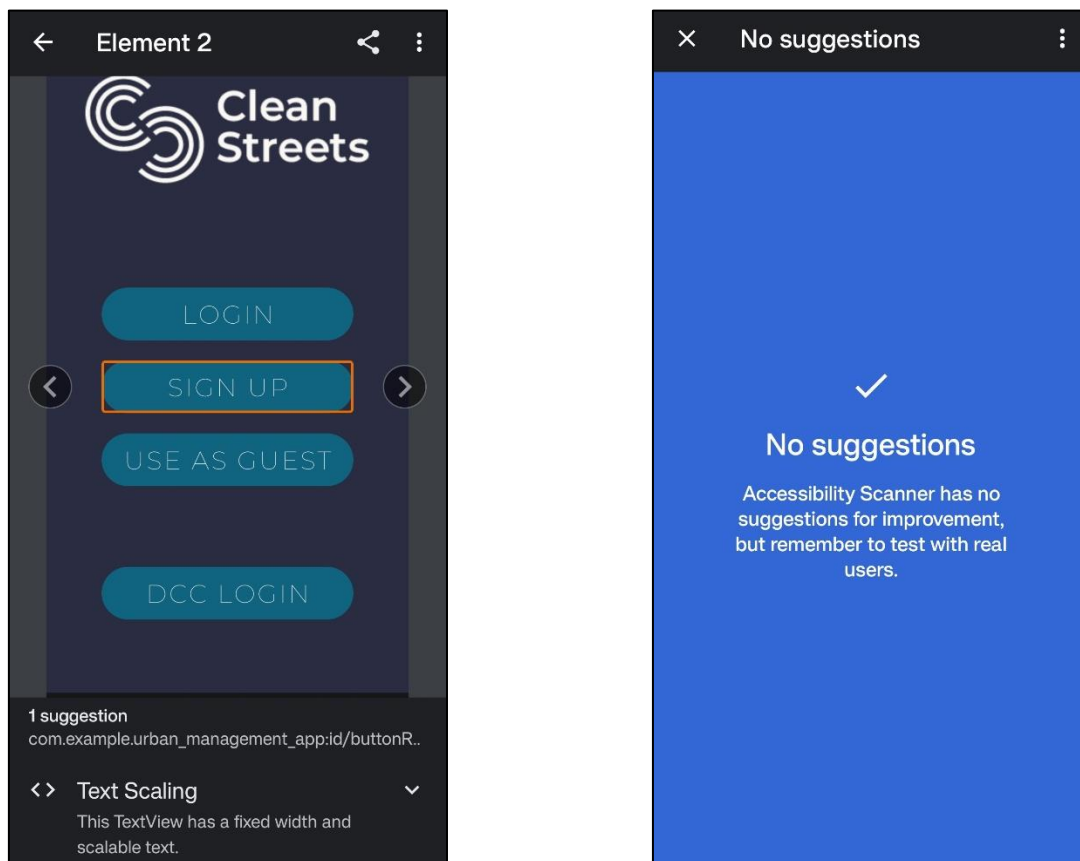


Figure 81: Google Accessibility Scanner results

5.4. Key Evaluation Learnings

The main aim of all evaluation approaches is to define the user's problems and explore any assumptions made in the development of the system. Several other qualitative approaches exist, and some of the following will also be explored as potential ways of getting better user feedback.

- **Shadowing:** This approach reduces risks before releasing the software, the approach is to monitor the differences between two environments, comparing and reducing the risk before implementing new functionalities.
- **Think-Aloud Protocol:** It is when a user speaks out loud what they are thinking while going through tasks on a system.
- **Co-Discovery Learning:** This is an adaptation of the Think-Aloud user test, where groups in pair think aloud to each other.

Having listened to all of the users during these evaluation tests, there was some interesting points brought up and questions raised that I had to answer for, and of course, evaluate in a developer sense. Some of the key learnings from the evaluation process include the following:

- *"I would like to see road names at least"* (Road names were added, while maintaining the minimal map feel)

- *“How do I go back?”* (iOS user unfamiliar with back buttons)
- *“Why does it take so long to upload?”* (Evaluation environment had weak Wi-Fi)
- Listening to each tester helped to make me understand the thought process of an end user, which was refreshing after months of programming focused development.
- The evaluation process opened areas for improvement, and helped to achieve project goals more efficiently.
- It allowed me to view “issues” as opportunities for development rather than obstacles.
- The process strengthens the ability to report on the project and provide useful information to improve future activities.

5.5. Conclusions

With the testing and evaluation plan defined, the app can be tested as soon as the features are sufficiently developed. Changes may be made to the design which will warrant more tests, but they can easily be added to the test plan. Once the app is evaluated, it can be altered if the results are unsatisfactory. The following chapter describes the prototype development process.

6. Conclusions and Future Work

6.1. Introduction

In this final chapter, the future work and outstanding issues will be discussed. It will act as a loose roadmap for the remainder of the project.

6.2. Issues and Risks

This subsection will break down some of the outstanding considerations that will need to be made within the development and scope of the system. Aspects of “Social Good” will be discussed.

6.2.1. Social Good

Sustainability

CleanStreets is a prime example of sustainable development, focusing on economic viability, environmental mindfulness, and social fairness (Mankoff, Kravets, & Blevis, 2008). Financially, it ensures both initial cost-effectiveness and long-term financial stability. Environmentally, *CleanStreets* carefully considers every step, promoting efficient resource use and responsible consumption through optimized routing. In terms of social impact, the app adopts a community-centric approach, minimizing adverse effects on well-being and community dynamics. *CleanStreets* inherently follows sustainability frameworks and their principles, making it a comprehensive, sustainable solution.

Legal

CleanStreets takes legal considerations seriously, addressing key standards to ensure a lawful operation. The app prioritizes user privacy following rules like GDPR (Greengard, 2018), which guarantees that user information is handled with care and consent. Additionally, *CleanStreets* respects intellectual property laws, ensuring its innovations are protected and respecting the rights of others. The app is designed to comply with relevant regulations governing its field, such as face blurring, showcasing a commitment to legal integrity. While this report may not detail every law, it emphasizes *CleanStreets*' dedication to operating ethically within legal boundaries. This not only reduces legal risks but also builds trust among users and stakeholders.

Accessible

“Mobile applications play an important role in many aspects of life. It is essential to be aware of the software development approaches that can support the design of accessible applications.”, (Zaina, 2022). CleanStreets is committed to accessibility, embodying a user-centered approach that goes beyond compliance. By adhering to standards outlined in the Web Content Accessibility Guidelines, the app ensures an inclusive experience for users of diverse needs. CleanStreets embraces accessibility requirements, considering the unique challenges faced by different user groups. The interface is designed with accessibility features, promoting a seamless and user-friendly experience for everyone, regardless of their abilities. The commitment to accessibility extends beyond mere compliance, reflecting CleanStreets' dedication to creating a digital space that truly caters to the needs of all users.

Security

CleanStreets prioritizes security with a tiered framework that aligns with industry standards, ensuring the protection of user data and the integrity of the application (Tayan, 2017). Adhering to comprehensive security frameworks, including measures for data security and authentication protocols, CleanStreets employs a multi-layered approach to safeguard user information. Rigorous encryption techniques are implemented to secure data both in transit and at rest, guaranteeing the confidentiality and privacy of user details. The authentication process is designed with precision, incorporating industry-best practices to validate user identities and prevent unauthorized access. CleanStreets stands as a pillar of security, fostering user trust through its commitment to stringent security frameworks and standards.

6.2.2. Additional Challenges

- **Application risk:** The connections between each layer is very important step in this project for the application to work successfully. The failure of connections may affect the final application performance.
- **Internet issues:** This application will be designed for a web application in which the internet connection is essential for the user to start the communication with the chatbot and the developer to deploy the updates.
- **Language gap:** This application is only available in English; however, Dublin is a multi-cultural city with many tourists coming from abroad, therefore this application can be further developed to include the other languages such: Irish, Spanish, French.

6.2.3. Self-reflection & Criticism

- **Personal development:** As this project has evolved over the past few months, I have become much more aware of the true sources of my project idea, particularly James Wilson and George Kelling's 'Broken Windows' Theory (Wilson & Kelling, 1982), which goes on to explain how *"Being forced to confront minor problems can heavily influence how people feel about their environment"*; this mindset, ultimately, drove the project forward in the name of creating more livable areas for everyone. Furthermore, I have gained a much deeper understanding of the social and infrastructural aspects of design and the mindset required to create a truly sustainable, accessible and functional city.

I have grown angry at the state of the city that we accept on a daily basis, and have educated myself online and by reading books and articles about sustainable, welcoming cities all over the world, from New York to Mumbai, from incredible people dedicated to transforming their own car-centric towns into inviting, beautiful areas that put people first. Janette Sadik-Khan, Former Commissioner of the New York City Department of Transportation, writes: *"Carving out exclusive bus lanes, paths for bicycles, expanded sidewalks and open spaces using remnant properties for small parks demonstrates what can be done with vision, strong technical and communication skills coupled with strong political leadership."* (Sadik-Khan, 2016)

Finally, I would like to mention *"Building the cycling city: the Dutch blueprint for urban vitality"* by (Bruntlett & Bruntlett, 2018) for the incredible stories and ideas that they shared throughout the book, that I feel like definitely impacted the design flow of the CleanStreets project, as well open my mind to incredible new ideas, such as the concept of how *"The ground floor (plinth) [of a building] makes up only 10% of the building, but determines 90% of the buildings contribution of the urban environment."* – And then asking myself why not apply this concept to the very ground that supports and anchors these buildings to the urban fabric?

- **Programming Development:** I have become comfortable talking about how my project actually functions, from frontend to backend and everything in-between, whereas before I began, I would only be able to talk about the various components individually. I believe that I have become a better programmer, and a better computer scientist because of this, as I faced so many challenges and standards to follow, such as GDPR (due to the handling of sensitive data), new coding concepts (i.e. accessing specific JSON entries

from an API call) and working with maps (Geographical Information Systems & routing).

I understand how important the decision is of choosing the correct tech stack while building a project, *“the choice of the technology stack is crucial to get right since it can make or break a project and is usually hard and expensive to change in the future.”*, (Martinsson & Svanqvist, 2022). I had to ask myself so many questions; Mobile or web? Free or monetized? Push notifications? Scalable database? And so on. I will definitely take the lessons I have learned from this development into future work.

- **Criticism of external resources:** Within the development lifecycle of the project, I have consulted many online resources for inspiration, samples and guidance. I have previously self-reflected, but I think that I also have some grounds to judge the sections of the project where I used external help. For example:
 - **Google Maps API** – Why is it so complicated to extract the polyline required for mapping the best route between two points?
 - **Android Studio** – Why can I not set the location of the device to wherever I want? This would help immensely in developing mapping applications in my opinion.
 - **MPAndroidChart** – Why is there so much unnecessary detail in the most basic graphs? I have to use so many additional parameters to get a clean, readable chart.

6.3. Plans and Future Work

This section will briefly discuss the plans that go beyond the scope of the project, including additional features, upgrades to the technology and integration with later versions of existing technology.

6.3.1. Gantt Chart

The Gantt chart segments the remaining targets for the project, broken down week by week. This is subject to revision and may be altered if necessary.

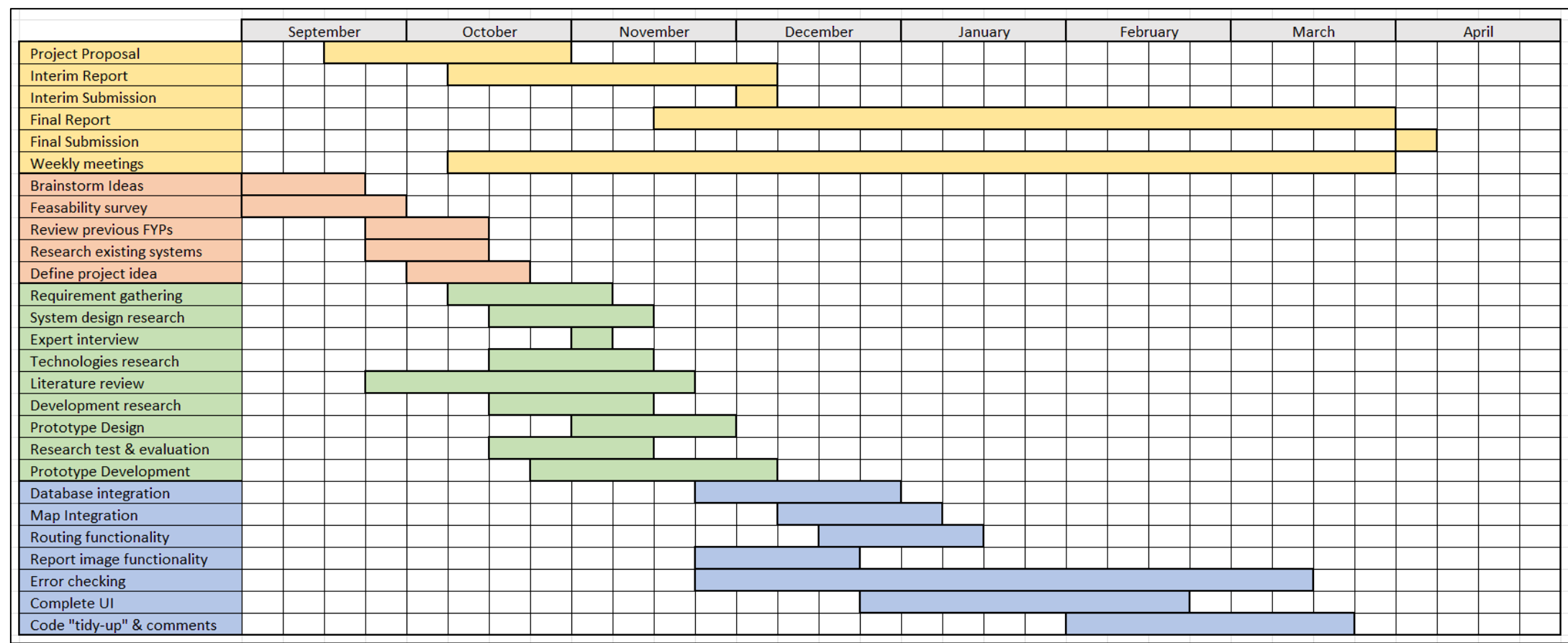


Figure 82: Gantt chart for the remainder of the CleanStreets project

6.3.2. Further Development

If development of the CleanStreets project was to continue, there are some key features that have been mentioned previously, that would have high priority in later versions of the application:

- **Image Processing:** This feature would be used to protect the end users from harmful or inappropriate imagery, as well as hiding the identities of people caught in the background of certain images. Other small aspects of Image Processing can be used to further enhance security like blurring address numbers and registration plates. If this feature was to be implemented, a library like **TensorFlow** or **OpenCV** would most likely be utilised. Additionally, there are apps, such as ObscuraCam¹ or Clouldinary² that could be used.

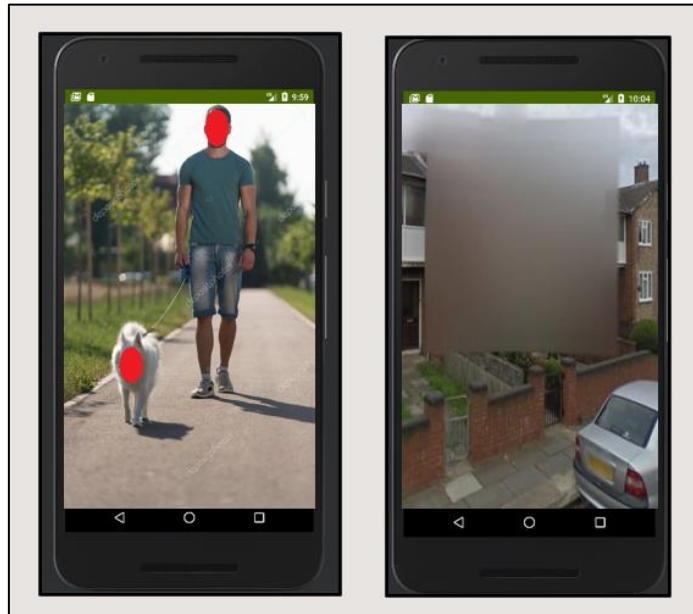


Figure 83: Future work - Image Processing

- **Cloud Management:** As the userbase grows, considerations will need to be made into expanding the capacity of the databases connected to the CleanStreets project. As a solo developer in this project, the funds are not available right now to scale the storage upwards or to implement cloud features like elasticity & better security. A potential Cloud architecture could be a MERN architecture - MongoDB, Express JS, React and NodeJS, with Python being the technology for routing components and data ingress. The React frontend could send HTTP requests to the Express JS REST API backend which could then query the Mongo Database for data which would be sent back to the React frontend. The Python data ingress and routing components could also update the data within the database.

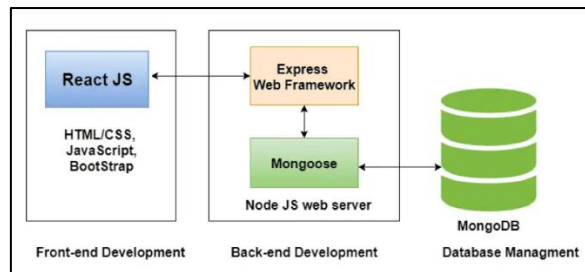


Figure 84: Future work - MERN Architecture

¹ <https://guardianproject.info/apps/org.witness.sscphase1/>

² https://cloudinary.com/blog/automatic_face_blurring_in_images_made_easy

- **Integration with Dublin City Council:** As this project evolves, the amount of sensitive data that will need to be processed, secured and translated to DCC systems will only increase. As it is an individual project, with some communication between developer and DCC, it is understandable that it could take months, if not years, before a genuine consideration to integrate the CleanStreets application is made.
- **Full notification support:** Users of the app deserve the peace of mind, and not to mention, the satisfaction of receiving the (hopefully!) good news that their report has been cleared by someone in their community or the DCC. Unfortunately, this version of CleanStreets does not have any notification feature. A quick fix would be sending emails to the users when their report's status' have been updated, this will be available in later versions.

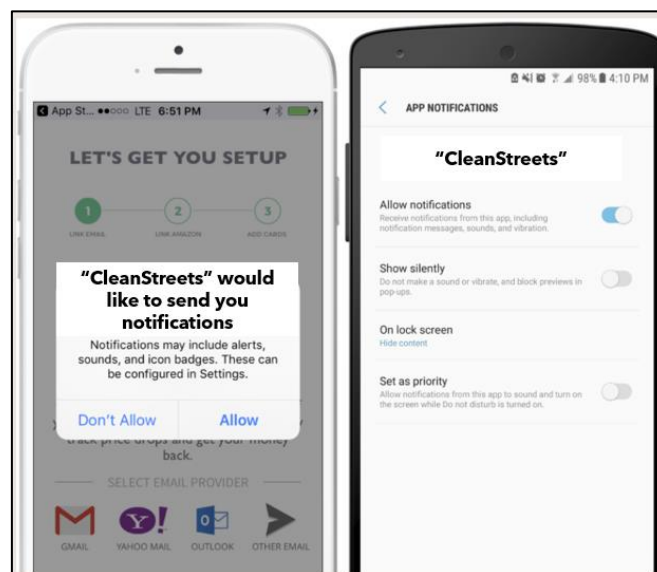


Figure 85: Future work - Full Notification Support

- **Full Accessibility features:** In future releases of the CleanStreets app, system-wide accessibility support is planned, and will be implemented on all pages on the app.

6.4. Conclusions

This chapter presented the direction in which the project is heading. It focused on three key aspects of the future work, first looking at the risks in the context of the social good,

secondly, a project plan is presented, and finally, considerations for up to 5 additional upgrades to the app across newer releases.

Finally, I am proud to have a complete project that I can call my own. Despite encountering many hurdles small and large, the final application works as intended, making it successful. Overall, a lot has been learned from the project experience, from initial research, design, developing, testing and evaluation of the system. The process introduced me to many new technologies, expanding my knowledge in many areas. While some aspects of this project were relatively tough, the knowledge and skills gained from overcoming them will definitely aid me greatly in future endeavours.

Bibliography

Figure 1 (n.d.). Retrieved from [universaldesign.ie](https://universaldesign.ie/what-is-universal-design/the-7-principles/): <https://universaldesign.ie/what-is-universal-design/the-7-principles/>

(n.d.). Retrieved from ProductPlan.com: <https://www.productplan.com/glossary/feature-driven-development/>

(n.d.). Retrieved from vtestcorp.com: <https://www.vtestcorp.com/blog/spiral-model/>

(2020, January 6). Retrieved from <https://www.irishtimes.com/news/social-affairs/dublin-s-north-inner-city-is-seriously-littered-1.4131214>

(2022, May). Retrieved from Institute Project Management: <https://www.projectmanagement.ie/blog/waterfall-methodology/>

Adekotujo, A. S. (2020, July). A Comparative Study of Operating Systems: Case of Windows, UNIX, Linux, Mac, Android and iOS. *International Journal of Computer Applications*.

Amazon AWS. (n.d.). *What is PostgreSQL?* Retrieved from [aws.amazon.com](https://aws.amazon.com/rds/postgresql/what-is-postgresql/): <https://aws.amazon.com/rds/postgresql/what-is-postgresql/>

Amazon. (n.d.). *What is a Lamp Stack?* Retrieved from <https://aws.amazon.com/what-is/lamp-stack/>

Apple Inc. (n.d.). *Swift*. Retrieved from <https://developer.apple.com/swift/>

Boehm, B. W. (1988). *A spiral model of software development and enhancement*.

Brilliant. (n.d.). Retrieved from <https://brilliant.org/wiki/traveling-salesperson-problem/>

Bruntlett, M., & Bruntlett, C. (2018). *Building the cycling city: the Dutch blueprint for urban vitality*. Island Press.

Carnegie Mellon University. (n.d.). Retrieved from <https://www.cs.cmu.edu/~biglou/resources/bad-words.txt>

Chatterjee, M. (2023, November 8). *What is A* Search Algorithm?* Retrieved from [greatlearning](https://www.mygreatlearning.com/blog/a-search-algorithm-in-artificial-intelligence/): <https://www.mygreatlearning.com/blog/a-search-algorithm-in-artificial-intelligence/>

Chodorow, K. (2013). MongoDB – The Definitive Guide 2e. In K. Chodorow, *MongoDB – The Definitive Guide 2e*.

Cresswell, J., & Miller, D. (2000). *Determining Validity in Qualitative Inquiry*.

Davies, A. (2007, January 22). Retrieved from <https://www.tandfonline.com/doi/full/10.1080/09644010601073564>

Flanagan, D. (2020). *JavaScript: The Definitive Guide. 7th Ed., O'Reilly Media*. Retrieved from <https://aip.scitation.org/doi/pdf/10.1063/1.168647>

Fowler, M. (2012). Patterns of Enterprise Application Architecture: . In M. Fowler, *Patterns of Enterprise Application Architecture*: . Addison-Wesley.

GCFGlobal. (n.d.). Retrieved from what is macOS?: <https://edu.gcfglobal.org/en/macosbasics/all-about-macos/1/#>

Google. (n.d.). Retrieved from Google Maps Platform: <https://developers.google.com/maps/documentation/directions/overview#:~:text=The%20Directions%20API%20is%20a,XML%2Dformatted%20directions%20between%20locations.>

Google ML Kit. (n.d.). *ML Kit*. Retrieved from Google ML Kit: <https://developers.google.com/ml-kit/guides#:~:text=ML%20Kit%20is%20a%20mobile,create%20brand%2Dnew%20user%20experiences.>

Gosling, J. (2005). In *The Java Language Specification, Third Edition*.

- Grant, K. J. (2018). In *CSS in Depth*. Manning.
- Greengard, S. (2018). Weighing the impact of GDPR.
- Greenwald, R. (2013). Oracle Essentials, 5th Edition. In R. Greenwald, *Oracle Essentials, 5th Edition*.
- Guerra, P., & Basani, M. (2023, April 25). Retrieved from <https://blogs.iadb.org/agua/en/smart-waste-technologies-where-are-we-and-where-are-we-going/>
- Hamilton, J., & Mullarkey, M. (2021, July 15). Retrieved from chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/<https://pubs.mumacasereview.org/2021/MCR-06-16-Hamilton-IoT-p1-25.pdf>
- Hřib, Z. (2019, October 14). Retrieved from themayor.eu: <https://www.themayor.eu/en/a/view/prague-introduces-smart-bins-to-save-energy-and-money-3527>
- Jo, H., & Baek, E.-M. (2023). *Exploring the dynamics of mobile app addiction: the interplay of communication, affective factors, flow, perceived enjoyment, and habit*. BMC Psychology.
- Juviler, J. (2022, September 14). Retrieved from HubSpot: <https://blog.hubspot.com/website/google-maps-api>
- Knöll et al., M. (2019). *Insights from Darmstadt, Germany*. Darmstadt.
- Lardinois, F. (2019, May 7). Retrieved from https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xllmNvbS8&guce_referrer_sig=AQAAANcvjOs9N414qfvY0OZ3008XVDfcH15P7iPp1zHgwLVQ-GveY2xdvRI51Qyt3cz5h7FsFC
- Lutz, M. (2013). In Lutz, M. (2013) *Learning Python, 5th Ed., O'Reilly Media*.
- Mankoff, J., Kravets, R., & Blevis, E. (2008). Some Computer Science Issues in Creating a Sustainable World.
- Martinsson, H., & Svanqvist, V. (2022). *Technology stack selection: Guidelines for organisations with multiple development teams*.
- Meike, G. (2021). Inside the Android OS. In G. Meike, *Inside the Android OS: Building, Customizing, Managing and Operating Android System Services (Android Deep Dive)*.
- Mullally, U. (2023, June 12). Retrieved from <https://www.irishtimes.com/opinion/2023/06/12/una-mullally-dublin-is-a-dirty-smelly-sticky-old-town-once-again/>
- Nagendra, B. L., & Agarwal, P. (2019). *Mobile application in municipal waste tracking: a pilot study of "PAC waste tracker" in Bangalore city, India*. The Journal of Material Cycles and Waste Management.
- Navone, E. C. (2020, September 28). Retrieved from freeCodeCamp: <https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/>
- Nidhra, S. (2012). Black Box and White Box Testing Techniques - A Literature Review. *International Journal of Embedded Systems and Applications*.
- Nielsen, J. (2000). *Why you only need to test with 5 users*. Nielsen, J.
- Ocampo, Y. (2022, October 10). Retrieved from <https://opengovasia.com/singapore-launches-smart-bins-to-boost-recycling/>

- Oinas-Kukkonen, H., & Harjumaa, M. (2008). *A systematic framework for designing and evaluating persuasive systems*. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.
- OpenCV. (n.d.). *About OpenCV*. Retrieved from OpenCV.org: <https://opencv.org/about/>
- OpenStreetMap. (n.d.). *About OpenStreetMap*. Retrieved from OpenStreetMap: https://wiki.openstreetmap.org/wiki/About_OpenStreetMap
- Pernice, K. (2017, November 12). Retrieved from <https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/>
- Purcell, M., & Magette, W. (2010, February 21). Retrieved from sciencedirect.com: https://www.sciencedirect.com/science/article/abs/pii/S0956053X10001078?casa_token=w13EQkp8E10AAA:JK8hu2VFZJSp0NqaQ5iNXdiPopReU5SGOhMan2Lg3jRp0XJke5aAo4xACUZ6Z5IsCrBUUldS9izw
- Royce, W. (1970). *Managing the development of large software systems: concepts and techniques*.
- Rychlý, M., & Tichá, P. (2008). *A Tool for Supporting Feature-Driven Development*.
- Sadik-Khan, J. (2016). *Sadik-Khan, Janette, and Seth Solomonow. "Streetfight: handbook for an urban revolution*. New York.
- Saher, R., Saleh, M., & Anjum, M. (2023). *Trash Collection System Integrating Human Collaboration with Technology*.
- sf.gov*. (n.d.). Retrieved from *sf.gov*: <https://sf.gov/sf311-mobile-app>
- Siever, E. (2009). *Linux in a Nutshell 6e*. In E. Siever, *Linux in a Nutshell 6e*.
- Skeet, J. (2019). In *C# in Depth, Fourth Edition*. Manning.
- Smart Prague*. (n.d.). Retrieved from *smartprague.eu*: <https://smartprague.eu/projects/mobile-application-my-prague>
- Snyder, C. (2003). *Paper prototyping: The fast and easy way to design and refine user interfaces*.
- Snyk*. (n.d.). Retrieved from *Snyk*: app.snyk.io
- Spiral Model*. (n.d.). Retrieved from *techtarget.com*: <https://www.techtarget.com/searchsoftwarequality/definition/spiral-model#:~:text=The%20spiral%20model%20is%20a,large%2C%20expensive%20and%20complicated%20projects>
- STAFF, W. (2019, December 11). Retrieved from <https://www.webdesignerdepot.com/2019/12/is-the-f-pattern-still-relevant-in-web-design/>
- Stevenson, D. (2018, September 24). *What is Firebase? The complete story, abridged*. Retrieved from Medium: <https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>
- Suruliraj, B., Nkwo, M., & Orji, R. (2020). *Persuasive mobile apps for sustainable waste management: a systematic review*. Aalborg, Denmark: Springer International Publishing.
- Tayan, O. (2017). Concepts and Tools for Protecting Sensitive Data in the IT Industry: A Review of Trends, Challenges and Mechanisms for Data-Protection. *International Journal of Advanced Computer Science and Applications* 8(2).

- TechTarget. (n.d.). *TechTarget: Agile, DevOps and software development methodologies: Spiral Model*. Retrieved from TechTarget: <https://www.techtarget.com/searchsoftwarequality/definition/spiral-model#:~:text=The%20spiral%20model%20is%20a,large%2C%20expensive%20and%20complicated%20projects>.
- TensorFlow. (n.d.). *TensorFlow*. Retrieved from TensorFlow.org: https://www.tensorflow.org/lite#:~:text=TensorFlow%20Lite*%20is%20an%20open,natural%20language%20tasks%2C%20and%20more.
- Tuama, D. Ó. (n.d.). *How to Use PHP in HTML*. Retrieved from codeinstitute.net: <https://codeinstitute.net/ie/blog/php-in-html/#:~:text=PHP%2C%20or%20Hypertext%20Preprocessor%2C%20is,Javascript%20code%2C%20and%20PHP%20code>.
- Unadkat, J. (2023, March 17). Retrieved from BrowserStack: <https://www.browserstack.com/guide/learn-software-application-testing>
- Vinney, C. (2021, July 30). Retrieved from <https://careerfoundry.com/en/blog/ux-design/universal-design-principles/#:~:text=The%20seven%20principles%20of%20universal%20design%20are%3A%20Equitable%20use%2C%20flexibility,space%20for%20approach%20and%20use>.
- Volle, A. (2023, November 15). *iOS*. Retrieved from Britannica: <https://www.britannica.com/topic/iOS>
- WebAim: Contrast Checker*. (n.d.). Retrieved from WebAim.org: <https://webaim.org/resources/contrastchecker/>
- Wegmann, A., & Genilloud, G. (2000). *The role of "Roles" in use case diagrams*. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Wilson, J., & Kelling, G. (1982). *psychologytoday.com*. Retrieved from <https://www.psychologytoday.com/us/basics/broken-windows-theory#:~:text=George%20L.,fully%20supported%20by%20subsequent%20research>).
- Zaina, L. A. (2022). *Preventing accessibility barriers: Guidelines for using user interface design patterns in mobile applications*. Journal of Systems and Software.

Figure 1: Illegal dumping in Dublin	8
Figure 2: Mind map from the early stage of development.....	9
Figure 3: The overall project represented using branches	11
Figure 4: Example of the MyPrague app (Hřib, 2019).....	15
Figure 5: BINgo smart recycling bin in Singapore (Ocampo, 2022)	16
Figure 6: SF311 App for San Francisco	17
Figure 7: 1-Tier Architecture	25
Figure 8: 2-Tier Architecture	25
Figure 9: 3-Tier Architecture	25
Figure 10: N-Tier Architecture	25
Figure 11: Routing example (Brilliant, n.d.)	28
Figure 12: The Waterfall Methodology diagram.....	34
Figure 13: The Spiral Model (vtestcorp.com, n.d.)	35
Figure 14: Feature-driven Development lifecycle (ProductPlan.com, n.d.)	36
Figure 15: Software Architecture of the system.....	37
Figure 16: Paper prototype of the Main Screen	38
Figure 17: Paper prototype of the Login & Registration Screens	38
Figure 18: Paper prototype of the Navigation Bar & Report List screen.	39
Figure 19: Paper prototype of the New Report Creation & Map Screens	39
Figure 20: Medium fidelity mock-ups in Figma; Main, Login and Register screens.	40
Figure 21: Medium fidelity mock-ups of Navigation Bar and Reports List	41
Figure 22: Medium fidelity mock-ups of New Report and Map Screens.....	42
Figure 23: Initial Use Case diagram.....	43
Figure 24: Extended Use Case diagram including the external connections & navigation menu.	43
Figure 25: Examples of issues I have faced in Dublin City.....	44
Figure 26: Car tyres destroyed the grass verge; Worn paint on kissing gates.....	45
Figure 27: Broken stone wall; Outdated Park paint.....	45
Figure 28: Issues faced in my locality; Overgrown bushes, defaced bin.	46
Figure 29: Heatmap of eye tracking data representing Nielsen's F-pattern (STAFF, 2019).....	51
Figure 30: Logo & typography design by Neuem (Marios Andreou)	54
Figure 31: Typography & marketing samples (Neuem)	55
Figure 32: Sample Accessibility page	56
Figure 33: Emulated "Simple Colours" feature	56
Figure 34: WCAG Contrast Checker results.....	57
Figure 35: Initializing variables for reports within the CleanStreets App	58
Figure 36: Initializing Firebase connections within the CleanStreets App.....	59
Figure 37: After error checking, the report class is created with the selected values	59
Figure 38: Taking user input to create an account with Firebase.....	59
Figure 39: Save new user to Database.....	60
Figure 40: Verify user login on subsequent visits	60
Figure 41: Built-in firebase password reset functionality	60
Figure 42: Anonymous sign in functionality.....	60
Figure 43: Request user-granted location permission.....	61
Figure 44: Connect to Google APIs using personal API key	61
Figure 45: Extracting report details based off marker position	61
Figure 46: Geocoding function, turning coordinates into human-readable addresses.....	62
Figure 47: Display report information from map marker	62
Figure 48: Plot the report and its details on the map.....	63
Figure 49: Default map style	64
Figure 50: Production map style	64
Figure 51: Extracting the current user location	64

Figure 52: Populate a new list of report coordinates	65
Figure 53: Using the list of coordinates to find the nearest report to the user	65
Figure 54: Google Maps API call.....	65
Figure 55: Extracting polyline from JSON response	65
Figure 56: Uploading a 'post' snippet	66
Figure 57: Sending 'post' to Firebase	66
Figure 58: Save user uid and 'reply'	67
Figure 59: Example screenshot of replies	67
Figure 60: Snippet of code showing the containsSwearWord function	68
Figure 61: Instantiation & usage of BadWordsFilter.....	68
Figure 62: Pre-processing of the data for analytics feature	68
Figure 63: isAdmin variable in user database	69
Figure 64: Admin only amend report option	69
Figure 65: thumbsUpIcon OnClick Listener.....	70
Figure 66: Report class thumbs-up getters and setters.....	70
Figure 67: Retrieve post_id and load existing replies	71
Figure 68: Reply error / swear checking	71
Figure 69: Early development navbar and register screens	72
Figure 70: Early development route-finding and report creation screens	72
Figure 71: Mid-development splash & reports map screens.....	73
Figure 72: Mid-development report creation & detailed report view screens	73
Figure 73: Late development main screen & Login	74
Figure 74: Late development detailed report and add report screens	74
Figure 75: Final product screenshots.....	76
Figure 76: Android Studio debugging QR code	79
Figure 77: Accessibility expert recommendation	82
Figure 78: Summary of security issues from Snyk	83
Figure 79: Three examples of project vulnerabilities	83
Figure 80: Allocation of resources high-risk issue.....	84
Figure 81: Google Accessibility Scanner results.....	85
Figure 82: Gantt chart for the remainder of the CleanStreets project.....	91
Figure 83: Future work - Image Processing.....	92
Figure 84: Future work - MERN Architecture	93
Figure 85: Future work - Full Notification Support	93